NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

A MODEL-BASED SYSTEMS ENGINEERING METHODOLOGY FOR EMPLOYING ARCHITECTURE IN SYSTEM ANALYSIS: DEVELOPING SIMULATION MODELS USING SYSTEMS MODELING LANGUAGE PRODUCTS TO LINK ARCHITECTURE AND ANALYSIS

by

Paul T. Beery

June 2016

Dissertation Supervisor Eugene P. Paulo

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2016	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE A MODEL-BASED SYSTEMS ENGINEERING METHODOLOGY FOR EMPLOYING ARCHITECTURE IN SYSTEM ANALYSIS: DEVELOPING SIMULATION MODELS USING SYSTEMS MODELING LANGUAGE PRODUCTS TO LINK ARCHITECTURE AND ANALYSIS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Paul T. Beery			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This dissertation contributes to model-based systems engineering (MBSE) by formally defining an MBSE methodology for employing architecture in system analysis (MEASA) that presents a comprehensive framework detailing the relationship between system architecture products and external models and simulations used to analyze system performance and feasibility. Specifically, the research combines the use of Systems Modeling Language (SysML) products and operational simulation models to support assessment of system requirements for systems engineering. The MBSE MEASA transforms operational needs into preferred system configurations through the analysis of detailed simulation models. The research does this by using designed experiments to generate architecture tradespace visualizations that highlight the impact that system design parameters, system-environment interactions, system operational implementation, and system component interactions have on system performance. The research demonstrates a procedure for iterations of the methodology when analysis suggests potentially impactful design, operational, or environmental variables (as well as potential interactions between those variables). The research develops and analyzes notional architecture products and simulation models of United States Navy mine warfare systems to demonstrate an application of the MBSE MEASA.

14. SUBJECT TERMS model-based systems engineering, Systems Modeling Language, system architecture, system analysis, modeling and simulation, mine warfare, MCM-1 Avenger, Littoral Combat Ship			**15. NUMBER OF PAGES** 283
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A MODEL-BASED SYSTEMS ENGINEERING METHODOLOGY FOR EMPLOYING ARCHITECTURE IN SYSTEM ANALYSIS: DEVELOPING SIMULATION MODELS USING SYSTEMS MODELING LANGUAGE PRODUCTS TO LINK ARCHITECTURE AND ANALYSIS

Paul T. Beery
B.A., Rutgers University, 2009
M.S., Naval Postgraduate School, 2011

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2016

Approved by: Eugene P. Paulo
Associate Professor of Systems Engineering
Dissertation Supervisor

Matthew G. Boensel Senior Lecturer Systems Engineering	Susan M. Sanchez Professor Operations Research
Kristin Giammarco Associate Professor Systems Engineering	Douglas H. Nelson Associate Professor Systems Engineering

Approved by: Ronald Giachetti, Chair, Department of Systems Engineering

Approved by: Douglas Moses, Vice Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This dissertation contributes to model-based systems engineering (MBSE) by formally defining an MBSE methodology for employing architecture in system analysis (MEASA) that presents a comprehensive framework detailing the relationship between system architecture products and external models and simulations used to analyze system performance and feasibility. Specifically, the research combines the use of Systems Modeling Language (SysML) products and operational simulation models to support assessment of system requirements for systems engineering. The MBSE MEASA transforms operational needs into preferred system configurations through the analysis of detailed simulation models. The research does this by using designed experiments to generate architecture tradespace visualizations that highlight the impact that system design parameters, system-environment interactions, system operational implementation, and system component interactions have on system performance. The research demonstrates a procedure for iterations of the methodology when analysis suggests potentially impactful design, operational, or environmental variables (as well as potential interactions between those variables). The research develops and analyzes notional architecture products and simulation models of United States Navy mine warfare systems to demonstrate an application of the MBSE MEASA.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

C2	Command and Control
DOD	Department of Defense
FFBD	Functional Flow Block Diagram
EFFBD	Enhanced Functional Flow Block Diagram
IDEF	Integrated Definition
INCOSE	International Council on Systems Engineering
JPL	Jet Propulsion Lab
LCS	Littoral Combat Ship
MBSE	Model-Based Systems Engineering
MEASA	Methodology for Employing Architecture in System Analysis
MCM	Mine Countermeasures
MILCO	Minelike Contact
MILEC	Minelike Echo
MIW	Mine Warfare
MOE	Measure of Effectiveness
NO/B	Nearly Orthogonal/Balanced
Non-MILCO	Non-Minelike Contact
Non-MILEC	Non-Minelike Echo
OOSEM	Object Oriented Systems Engineering Method
RI&N	Reacquisition, Identification, and Neutralization
SysML	Systems Modeling Language
SYSMOD	Systems Modeling
UML	Unified Modeling Language

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This dissertation defines a model-based systems engineering (MBSE) analysis methodology that links system architecture products with external models and simulations to analyze system performance. Current MBSE research has focused largely on the definition and formalization of Systems Modeling Language (SysML) products and diagrams, with insufficient definition of how SysML can be used to analyze the system. To address this gap this research proposes a MBSE method to link architecture models to analysis models. The MBSE MEASA integrates system architecture and the system analysis domains and maintains traceability, both forwards and backwards, from the system requirements to the system performance results. The MBSE MEASA leverages existing methods for designing, constructing, and analyzing large-scale simulation experiments to determine the drivers of system performance. The MBSE MEASA demonstrates a procedure for iteration of the methodology, based on analysis results, to integrate impactful design, operational, and environmental variables (as well as potentially impactful interactions between those variables) into subsequent SysML products.

Current direction of MBSE research devotes substantial energy to the definition of SysML diagrams, to document system architecture views from a functional and physical perspective and define an executable procedure for evaluating the consistency and correctness of the those system architectures. Ryan, Shahram, and Mazzuchi (2013) provide an overview of existing MBSE methods, frameworks, and standards, which highlights the broad range of current MBSE applications. They demonstrate that SysML diagrams can describe a system comprehensively, in terms of requirements, functions and physical components.

This focus on utilizing SysML products to define a system and to analyze the performance of that system extends to industrial applications of MBSE. Leaders in the engineering field, such as the International Council on Systems Engineering (INCOSE), IBM, and Vitech have developed MBSE methodologies. A thorough review of these methodologies shows that they share three major goals: definition of appropriate system

functions based on stakeholder identified system requirements; definition of the set of potential system physical components; and allocation of physical components to system functions (to be checked for consistency through a defined operational architecture). These methodologies are extremely effective at demonstrating whether a given set of physical components is capable of performing a given set of system functions through analysis of these executable allocated architectures. The MBSE MEASA developed in this dissertation expands the utility of these methodologies by prescribing how functional and physical architectures can be used to define external performance models which allow for examination of system performance in greater detail (by examining a large number of system design variables, environmental variables, and operational variables).

This dissertation recognizes that recent research has been largely segmented, with substantial developments occurring in functional and physical architecture development (the System Architecture Domain) and other developments occurring in modeling and simulation and system analysis (the System Analysis Domain). This research focuses on a revised approach for the integration of those domains. Specifically, this dissertation develops a comprehensive framework for the development of SysML based system architecture products and uses those products as the basis for the development and analysis of detailed external simulations. This allows the MBSE MEASA examine system performance in detail and to specify a procedure for iteration of the methodology from detailing system analysis results to subsequent system architecture products that is unique in the current literature.

The MBSE MEASA is based on the analysis methodology developed in MacCalman (2013), which presented a sequenced analysis approach demonstrated though analysis of early stage ship design. That work was expanded by MacCalman, Kwak, McDonald, and Upton (2015) to include architectural representations and to analyze other system design problems. The MBSE MEASA expands on that approach by formally prescribing the architectural representations that enable the implementation of more detailed external models and simulations (and associated system analysis). The MBSE MEASA is a five step process, each of which defines a sequence of activities and products that ensures traceability from stakeholder input to system solutions. The MBSE

MEASA prescribes a procedure for updating future iterations of existing architecture products and informing subsequent stakeholder communications. The MBSE MEASA also outlines a standardized format for information capture and model development that ensures that any changes to system configurations can be rapidly introduced into system architecture products and implemented in external system models.

The first three steps of the methodology are Requirements Analysis (Step 1), Functional Architecture development (Step 2), and Physical Architecture development (Step 3). Requirements Analysis defines the system in terms of its Real Environment as well as an initial set of Design-To-Specifications. Those requirements are the basis for Functional Architecture development, which defines the system in terms of the functions that the system must perform as well as the ordering and dependencies of those functions. This facilitates development of an initial set of system design parameters as well as measures of effectiveness and serves as a guideline for Physical Architecture development. This provides a mapping of the relationship between system components and the activities performed by each component. Together, these requirements and architectures support the final two steps of the methodology, Model Definition (Step 4) and Model Analysis (Step 5) and ensure consistency between model types (operational, physical, and cost models). Analysis of those models identifies preferred system configurations, which are based on based on operational, physical and cost models, which are based on functional and physical architecture, which are based on a stakeholder specified set of requirements. In this way, the MBSE MEASA ensures traceability from detailed system analysis to system stakeholder identified requirements, and establishes a mechanism for discussions between system architecture experts (Steps 1–3) and system analysis experts (Steps 4–5). This creates a unique opportunity for iteration of the methodology, where the results of detailed system analysis can be integrated directly into subsequent iterations of the methodology.

The dissertation applies the MBSE MEASA to a U.S. Navy mine warfare system. The dissertation builds a full set of SysML products, defines the requirements, activities, event sequences, use cases, states, and physical components that define a notional mine warfare system (the term “notional” is used to emphasize that there are assumptions and

limitations associated with the models that do not take away from the demonstration of the MBSE MEASA but limit the applicability of the analysis results to "actual" mine warfare systems). Two discrete event simulations are built based on the activities and components defined in the SysML products. The first simulation examines the operational effectiveness of the MCM-1 Avenger and its support systems, the current mine countermeasure (MCM) system for the U.S. Navy. The second simulation examines the operational effectiveness of the Littoral Combat Ship (LCS) and its support systems, the future MCM system for the U.S. Navy. Each simulation examines alterations to system configurations, system design parameters, system operational factors, and environmental factors (due to differences in operational employment, 51 factors are examined for the MCM-1 Avenger configurations and 32 variables are examined for the LCS configurations). That analysis results in several major findings. First, the operational performance of both the MCM-1 Avenger configurations and the LCS configurations is most significantly impacted by the number of passes that each system conducts through a minefield. Second, the probabilities of detection, classification, identification, and neutralization all have a substantial impact on system performance. The research also presents several alternative tradespace visualizations that define sets of system configurations that perform best with respect to four effectiveness measures (mine detection, mine clearance, operational duration, and operational cost).

This dissertation develops a MBSE MEASA contributes to the existing literature by linking architectural descriptions to analysis. The MBSE MEASA defines a procedure for the utilization of architecture products as the basis for detailed system models. Model analysis subsequently defines a more complete set of system requirements. That more complete set of system requirements informs subsequent iterations of system architecture products. Adherence to the MBSE MEASA ensures that a full set of SysML products describes the system in terms of system design parameters as well as the environmental and operational factors that may impact system performance. Analysis of external simulation models based on architecture products determines the system design parameters that have the greatest impact on system operational effectiveness, system design, and system cost. System tradespace analysis can then be used to identify a feasible set of system design parameters. Because the MBSE MEASA bases the

development and analysis of external models and simulations on detailed architectural descriptions, adherence to the MBSE MEASA ensures that set of system design parameters is traceable to a set of system requirements, as described using SysML architecture products. This facilitates rapid iteration of the process and integration of analysis results back into architecture products. The MBSE MEASA ensures traceability between system model analysis results and system requirements and establishes defined, defensible linkages between system architecture products and system analysis products. These traceable linkages facilitate interaction and discussion with system stakeholders to improve the design and analysis of systems.

LIST OF REFERENCES

MacCalman, Alexander D. 2013. "Flexible Space-Filling Designs for Complex System Simulations." Ph.D. Dissertation, Naval Postgraduate School.

MacCalman, Alex, Hyangshim Kwak, Mary McDonald, and Stephen Upton. 2015. "Capturing experimental design insights in support of the model-based systems engineering approach." In *Procedia Computer Sciences*: Vol. 44, edited by Jon Wade and Robert Cloutier, 315–324. Hoboken, NJ: Elsevier.

Ryan, Jessica, Shahram Sarkani, and Thomas Mazzuchi. 2013. "Leveraging Variability Modeling Techniques for Architecture Trade Studies and Analysis." *Systems Engineering* 17 (1): 10–25.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Lindsay and Olivia, thank you for being my best friends.

Gene, I cannot imagine that anyone could possibly be more responsible for someone else's success than you have been for mine. Thank you.

Matt, Kristin, Doug, Susan, I know that serving on a dissertation committee is often a thankless job, so I wanted to say thank you for taking your time to be patient with me and ensure that I produced the best possible dissertation.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The current challenges facing U.S. Department of Defense (DOD) system development are the primary motivation for this dissertation. In particular, because system development relies heavily on the creation of both system architecture products and system models, this dissertation develops an analysis methodology that establishes a link between those system architecture models and system analysis models. This analysis methodology acknowledges current accepted standards in systems engineering and model-based systems engineering, and leverages current research and architecture products to support the methodology.

From a more general perspective, the motivation for this research is a speech made in in April 2013 by U.S. Secretary of Defense Chuck Hagel reviewing the effectiveness and expense of DOD systems. Secretary Hagel stated, "We need to continually move forward with designing an acquisition system that responds more efficiently, effectively, and quickly to the needs of troops and commanders in the field" (Hagel 2013, 1). Secretary Hagel used this statement to stress that current DOD systems are often more expensive and more technologically risky than originally planned, and therefore future systems must be defined, planned, analyzed, and constructed with a focus on ensuring that those systems "do not continue to take longer, cost more, and deliver less than initially planned and promised." Implicit in Secretary Hagel's speech is that, while improvements to system development must ensure that DOD systems do not take too long, cost too much, and deliver too little, the system development process exists specifically because DOD systems necessarily have long development times, high costs, and high levels of complexity. These challenges have resulted in an increased DOD focus on the role of systems engineering in the system development process, and this has focused this research to demonstrate a linkage between system architecture products (which define what a system is intended to do as well as the physical components that will define the system) and system analysis products (which assess how well the system actually meets operational effectiveness standards).

This research presents a systems engineering analysis methodology that assists in making impactful engineering decision for large scale, complex systems. In particular, this research focuses on the appropriate development and definition of system architecture and system analysis models. Furthermore, given the complicated nature of the systems of interest (particularly the large number of system components and system component interfaces), this research focuses largely on simulation models due to their ability to consider a large number of input variables and a large number of operational scenarios in a repeatable, controlled environment. This frames the primary research goal as development of a model-based systems engineering analysis methodology specifically tailored to develop traceable system architecture products used to guide simulation model development to support system level decisions. Formalization of this methodology uniquely defines an iteration procedure for integration of analysis results into future iterations of architecture products. Accordingly, this dissertation develops an analysis methodology that supports production of complete system requirements via definition of appropriate linkages between system architecture and system analysis models.

B. RESEARCH FOCUS AND SUMMARY

The International Council on Systems Engineering (INCOSE) definition of systems engineering is instructive when first considering the development of an engineering analysis methodology. INCOSE defines systems engineering as, "an interdisciplinary approach and means to enable the realization of successful systems. It is focused on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal" (SE Handbook Working Group 2011, 6). While this definition suggests that systems engineering may be useful to support system development, INCOSE's definition of model-based systems engineering (MBSE) is even more instructive. INCOSE defines MBSE as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phase" (Technical Operations, INCOSE 2007, 15). The

definitions are largely similar, but the slight differences between INCOSE's definitions of systems engineering and MBSE are important. While both definitions emphasize that systems engineering should support requirements, design, analysis, and verification and validation activities, the MBSE definition specifies a mechanism by which this support is realized, specifically the "formalized application of modeling." This subtle difference more clearly establishes why MBSE is appropriate to support system development. MBSE intends to formally apply modeling and simulation to support definition of system requirements, system design, system analysis, and system verification and validation. System development necessarily leans on models (specifically simulation models) for support. While various MBSE methodologies exist, it is necessary to define an MBSE MEASA specifically tailored to analyze large scale, complex systems using external models and simulations, such as those developed in MATLAB/Simulink, ExtendSim, MASON, SimPy, AnyLogic, NetLogo, iThink or other simulation software packages. The current focus of MBSE research and methodology development necessitates this additional clarification regarding the use of "external models and simulations."

Within the domain of MBSE, substantial effort has been spent on creation of a standardized system architecture modeling language, the Systems Modeling Language (SysML). Bjorkman, Sarkani, and Mazzuchi (2013) recognize that, while development of SysML is important, there is limited research into MBSE system performance analysis. Specifically, they state, "although MBSE approaches have much promise for improving existing systems engineering processes, to date not much attention has been paid regarding the role of test and evaluation" (15). Similarly, INCOSE, IBM, Vitech Corporation, and NASA, have developed MBSE methodologies that demonstrate the value of utilizing SysML as an enabler of MBSE (those methodologies, along with several other preeminent MBSE methodologies, are reviewed in detail in Chapter II). While the creation of SysML and the development of MBSE methodologies have established an excellent framework for the development of clear, consistent system models, that research has focused primarily on development of system architecture models and has largely ignored the need to link system architecture products to detailed external models and simulations.

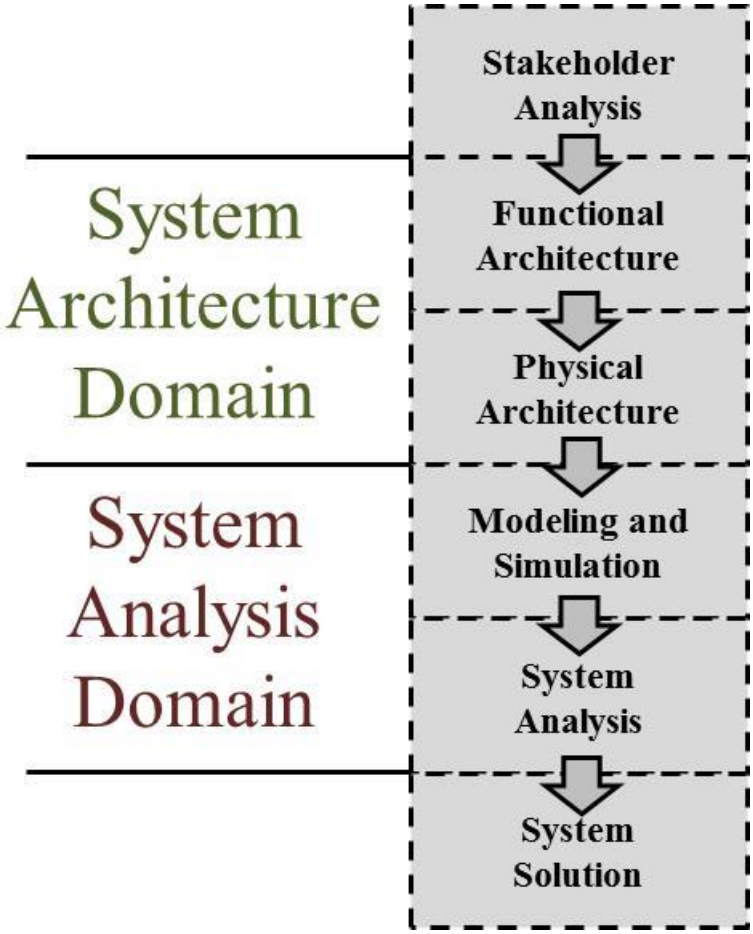
Examining the goals of each of the leading MBSE methodologies highlights a limitation to the current state of the art of MBSE. IBM Harmony for Systems Engineering satisfies three objectives: identification of system requirements; identification of system models; and allocation of system functions/states to system structure (Hoffman 2011). INCOSE's Object Oriented Systems Engineering Method (OOSEM) supports similar goals: understand system specifications; improve integration between physical systems and software systems; and facilitate system, element, and component reuse (INCOSE 2011). Vitech's MBSE Methodology links four domains of systems engineering, progressing from requirements to functions to physical elements to design validation and verification (Vitech Corporation 2011). Each of these methodologies has the same focus, they link stakeholder input to system functions and system physical components. Subsequently, they develop and execute allocated architectures, which map system physical components to system functions, which are used to validate and verify system physical design. While these methodologies comprehensively describe systems, they fall short in their analysis of system performance. Specifically, the use of allocated and executable architectures to verify and validate system design is insufficient to completely analyze system performance because they are incapable of completely examining a system in terms of the interactions between the system and the environment, the potential impact of alterations to system operation and implementation, as well as the interactions between system components. Detailed external models are required to completely examine these aspects of system performance.

Recent MBSE research, in particular Acheson, Dagli, and Kilicy-Ergin (2013), Cao, Liu, and Paredis (2011), Giammarco and Auguston (2013), Huang (2011), Huang, Ramamurthy, and McGinnis (2007), and Sitterle, Freeman, Goerger, and Ender (2015) has focused on expanding the conceptualization of MBSE beyond the development of system architecture models to the development of physical system models and operational system models. That work has, often necessarily, restricted model development and analysis to a single demonstration of either a physical system model or an operational system model. Recent analysis work, in particular Tolk and Hughes (2014) and MacCalman, Kwak, McDonald, and Upton (2015), has demonstrated the potential

utility of examining both physical system models and operational system models concurrently. The MBSE MEASA formalizes a comprehensive approach that details the transition from the current focus of MBSE (creation and examination of detailed system architectures) to the creation of system architectures that serve as the basis for the development and analysis of detailed physical system models and detailed operational system models. Further, the MBSE MEASA demonstrates the iteration of the methodology from the analysis of those physical and operational models into the previously developed system architecture models. Because the physical and operational models allow for a more detailed examination of system design, operational, and environmental variables, the MBSE MEASA is therefore able to prescribe an appropriate integration technique for introducing impactful variables of any kind into subsequent iterations of the system architecture. This provides a more detailed, integrated formalization of the use of system architecture to support system analysis (and the use of system analysis to support subsequent iterations of the system architecture) than is possible using any existing method.

An examination of the systems engineering process highlights the utility of the MBSE MEASA. Figure 1 provides an overview of a single iteration of an idealized, generic systems engineering process (Chapter II presents a detailed review of the leading systems engineering process models).

Figure 1 Single Iteration of a Generic Systems Engineering Process



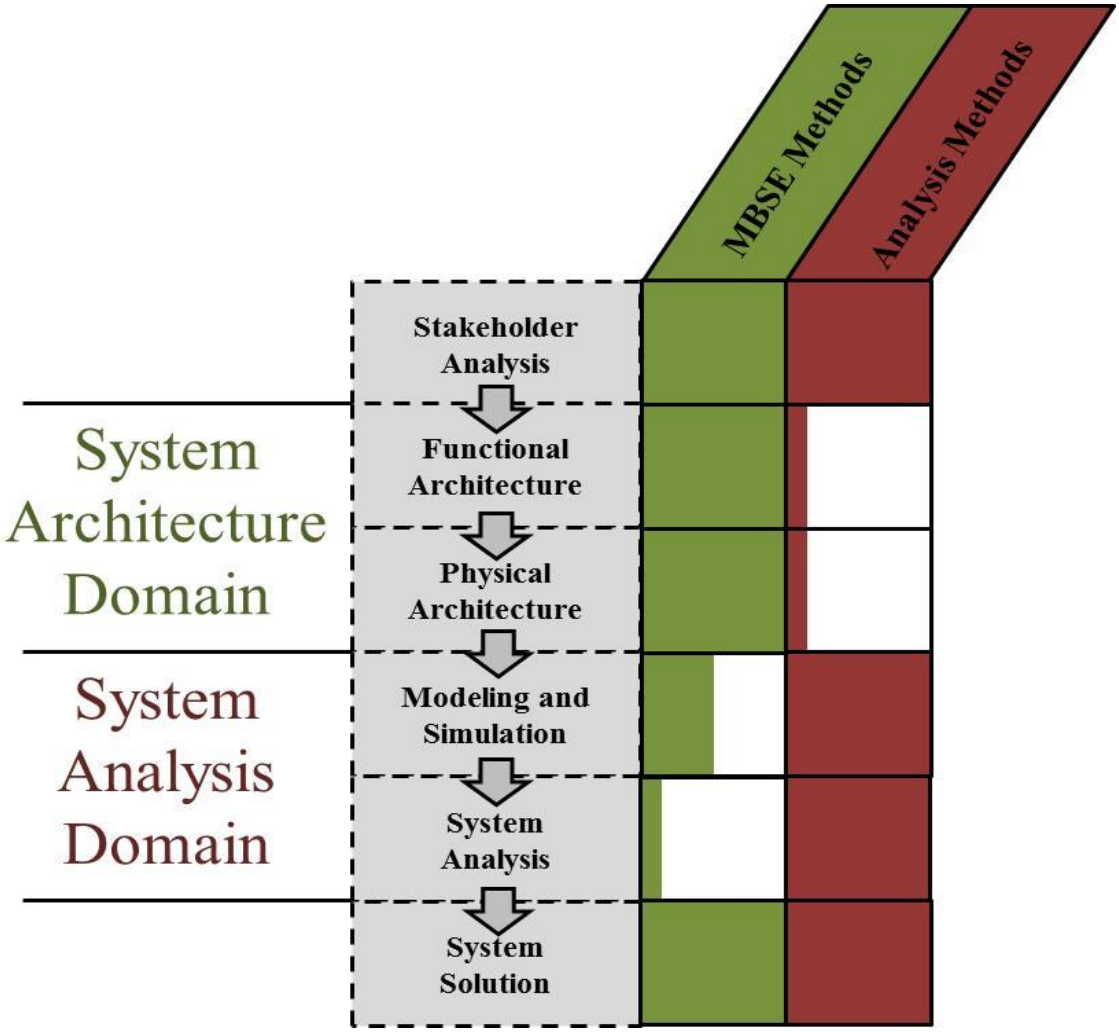
As a point of clarification, defining the terms "system architecture" and the "system architecture domain" in the context of this research is valuable. Perhaps the most widely read systems architecting textbook, *The Art of Systems Architecting* by Maier and Rechtin (2009), includes an Appendix devoted solely to producing a definition of system architecture. While this Appendix does not actually produce a clear, concise definition, it does identify several unifying characteristics of system architectures. It notes that system architectures identify and organize fundamental system components, relationships, interfaces, processes, constraints, and behaviors. In particular, architecting creates concrete objects, which are traditionally lists of components, relationships, interfaces, process, constraints, and behaviors. Development of these products is considered development of "system architectures" within the context of this research. Likewise, research and work supporting the creation of system architectures, from refinement of

system requirements to development of specifically defined architecture products, is considered work in the system architecture domain. Emes et al. (2012) examine the relationship between systems engineering and system architecture and likewise do not arrive at a complete definition of systems engineering or system architecture, but do develop a similar description of the role and purpose of systems architecting and system architectures. Accordingly, those unifying characteristics of system architecture guide this research.

Figure 1 does not completely describe all of the detail necessary to capture the systems engineering effort typically conducted throughout system development; rather it presents a generic representation of the commonly implemented systems engineering process. Sequentially, after a group of stakeholders is identified and an initial set of system requirements is developed through interaction with those stakeholders, system architectures are constructed, where a functional architecture specifies what the system must do in order to satisfy the developed requirements and the physical architecture specifies what system components are necessary to perform the functions identified in the functional architecture. Subsequently, system models and simulations are built and exercised, and the analysis of the outputs of those models and simulations develops a set of potential system solutions. The linkage of these processes, as well as iteration between the processes and subsequent iterations of the complete process, ensures that any recommended system solutions are based on previously conducted system analysis, which is based on system models and simulations, which are based on previously developed system architectures, which represent stakeholder needs. Tolk and Hughes (2014) advocate this defined, traceable process, stating “SE processes need to be aligned and synchronized to support a variety of technical team members and stakeholders and all phases of the life cycle of a system” (38). While adherence to the process is valuable, substantial domain specific expertise is required to conduct research that expands the scope and utility of any stage of the systems engineering process. Within MBSE, this problem has been compounded by the need to formally define the characteristics and utility of SysML. Due to the desire to gain acceptance for SysML as a standardized modeling language, the MBSE community has devoted substantial research time into the

analysis of the utility of SysML products and demonstration of the value of SysML development within the architecture domain. Figure 2 presents a graphical description of recent research in both the system architecture domain and the system analysis domain. A more detailed assessment of recent work conducted in the MBSE and analysis communities is presented in Appendix A to substantiate Figure 2.

Figure 2 Current MBSE Research Focus



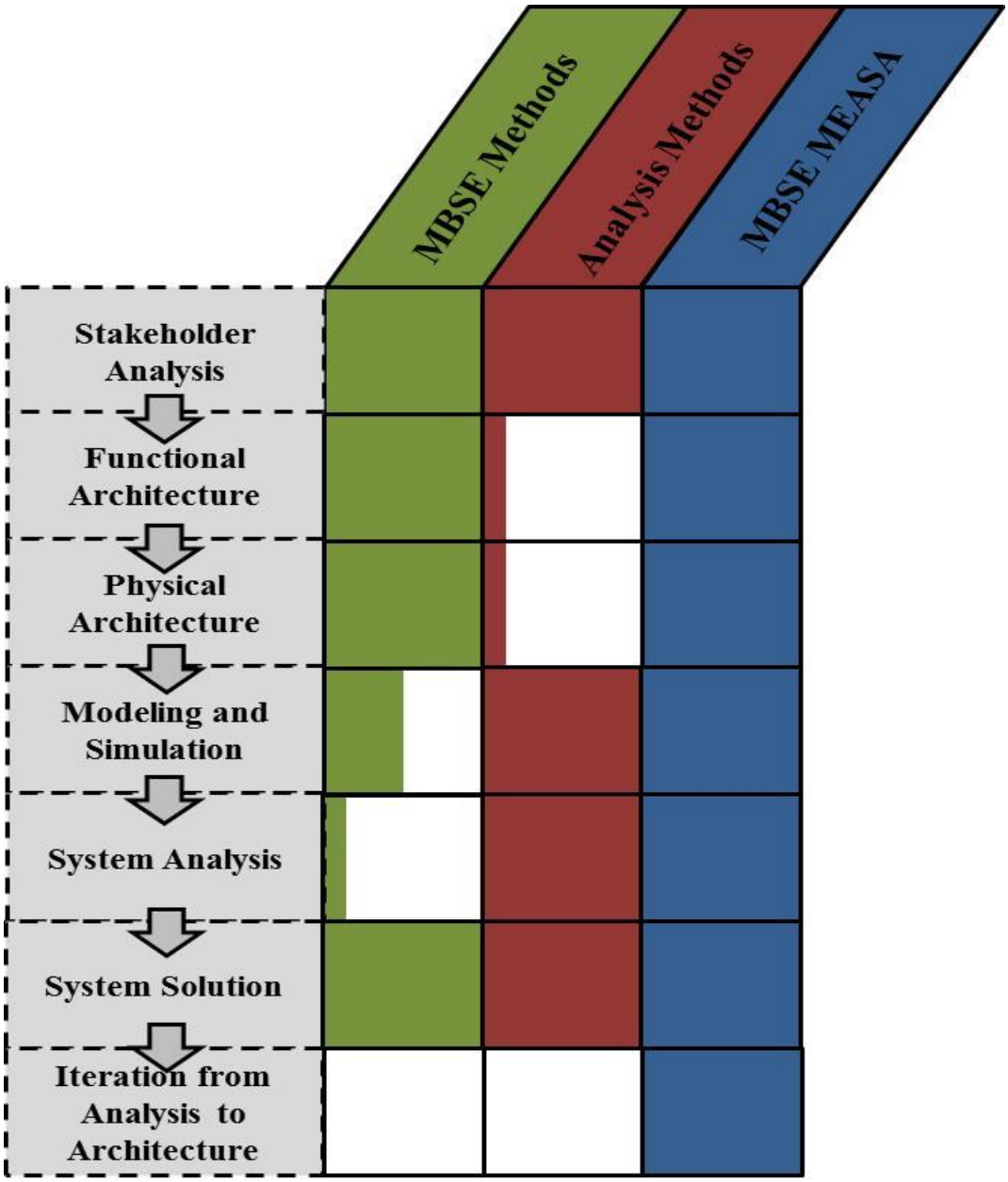
Note that Figure 2 suggests that the majority of the MBSE Methods research has focused in the system architecture domain. This aligns with the focus of each of the leading MBSE Methodologies, although only one of these MBSE Methodologies restricts

the external modeling to the architecture. Recent work by Acheson, Dagli, and Kilicy-Ergin (2013), Bataresh and McGinnis (2012), Cao, Liu, and Paredis (2011), Giammarco and Auguston (2013), Haveman and Bonnema (2015), Huang (2011), Huang, Ramamurthy, and McGinnis (2007), Neches and Madni (2013), Sitterle, Freeman, Goerger, and Ender (2015), and Wang and Dagli (2011) has expanded the scope of MBSE research to consider the creation of external models, as well as more detailed analysis. Other MBSE development research that use SysML architecture, and incorporate operational or environmental variables into external parametric or spreadsheet models, includes the work of Bjorkman, Sarkani, and Mazzuchi (2013), Cao, Liu, Fan, and Fan (2013), Carson and Sheeley (2013), Fisher (2013), Kim, Fried, Menegay, Soremekun, and Oster (2013), Ross (2003), Ross, Stein, and Hastings (2014), and Ryan, Sarkani, and Mazzuchi (2013). Similarly, there has been substantial work in the analysis community on the development of detailed analysis and tradespace visualization techniques that can be used to support system development. Lucas, Kelton, Sanchez, Sanchez, and Anderson (2015) describe the state-of-the-art in simulation modeling and analysis for addressing complex problems, and Sanchez, Lucas, Sanchez, Nannini, and Wan (2012) demonstrate the utility of such an approach to aid development and analysis of unmanned aerial vehicles. MacCalman, Kwak, McDonald, and Upton (2015) use this approach to develop and analyze operational models of an Army unit based on architecture models. While these efforts provide many useful demonstrations of the potential utility of linking architectural and simulation models, system architecture developers who are unfamiliar with simulation modeling may benefit from a more detailed description of that linkage.

The MBSE MEASA is most closely related to that of MacCalman (2013), who describes an MBSE analysis methodology for ship design (see also MacCalman, Beery, and Paulo working paper); the language in that methodology is adapted for more general system design problems in Chapter III. MacCalman and other co-authors subsequently go on to use this methodology, in conjunction with the integration of SysML and their external simulation models, for other system design problems (MacCalman, Kwak, McDonald, and Upton 2015). The MBSE MEASA provides a comprehensive formalization of the use of architecture models (the current focus of MBSE) as a basis for

the development of physical system models and operational system models. Further, the MBSE MEASA uniquely specifies how the results of the analysis of those models can be integrated back into future iterations of the system architecture. Figure 3 presents a visualization of the expected utility of the MBSE MEASA.

Figure 3 MBSE MEASA Intended Utility



The MBSE MEASA contributes to MBSE in the areas of system architecture and system analysis in such a way that the distinction between the two domains is no longer necessary. The MBSE MEASA formally defines the use of architecture to support analysis (and vice versa) to ensure that behaviors represented in the models and simulations created in the System Analysis Domain can be traced to functions prescribed in the System Architecture Domain. Similarly, it ensures that the system configurations and performance standards established in the physical architecture are consistent with the systems and system components created in any external models and simulations. The MBSE MEASA uses SysML products to formally describe these relationships, which enables integration and iteration of the process in a unique fashion. Because the MBSE MEASA creates dynamic architecture products (using SysML) as the basis for detailed system physical and operational models, the MBSE MEASA can incorporate a wide range of potential outcomes (specifically impactful design, operational, and environmental variables, as well as interactions between those variables) back into the system architecture products. The MBSE MEASA provides a comprehensive framework for the creation of system architecture products, the creation of external simulation models, and the iteration of the systems engineering process beyond the capabilities of any existing systems engineering approach.

This chapter presents a general description of the dissertation contribution, as well as motivation and relevant background information. This research demonstrates in Chapter II that the existing MBSE methodologies align closely with that process of creating products that describe the system of interest from architecture perspective, but do not provide a mechanism for detailed analysis of system performance using external simulation based on those architecture products. Chapter II also reviews recent developments in MBSE and simulation analysis to position the utility of the MBSE MEASA in terms of recent literature. In Chapter III, this research presents an analysis methodology that expands that architecture focus of current MBSE methodologies by identifying the appropriate usage of those descriptive architecture products to create external models and simulations that facilitate more in-depth exploration and analysis of system performance. Chapter IV subsequently demonstrates the application of this new

analysis methodology through a study of a notional U.S. Navy mine countermeasures system. Finally, Chapter V presents conclusions and recommendations for future research.

C. PROBLEM STATEMENT

This dissertation develops an MBSE MEASA for analyzing large scale, complex systems through operational simulations and system synthesis models. Current industrial MBSE research focuses on appropriate definition of functional, physical, and allocated architectures through the use of SysML products. Academic research has expanded that architecture focused approach by developing external models and simulations based on system architectures. Current system analysis research successfully applies models and simulations both with and without the use of detailed system architectures. This research presents a comprehensive framework that expands the applicability of the state of the art of MBSE by establishing traceability from detailed architectures to detailed external models (and back again). This research demonstrates how system architecture products developed in SysML can support a methodology for conducting detailed system analysis, and how analysis results can be integrated into subsequent iterations of system architectures. This new analysis methodology is demonstrated through a notional analysis of the operational performance and feasibility of a future United States Navy mine warfare system.

D. RESEARCH SCOPE AND ASSUMPTIONS

Systems engineering emphasizes the importance of creation of system architectures. Systems engineering also recognizes the important role that simulation and modeling can play in system testing. This research focuses largely on ensuring proper linkage between defined system architecture products and system simulation models. MBSE uses these simulation models to inform many aspects of a system. This research presents a revised approach to properly establish a linkage between the operational effectiveness of a system and its functional and physical characteristics. Specifically, the MBSE MEASA advocates a simultaneous investigation of both operational effectiveness and system feasibility through simulation. After simulations in each of these areas are

developed, executed, explored using designed experiments, and analyzed, predictive surrogate models are developed that link system performance characteristics to both operational effectiveness and system characteristics. This dissertation formalizes an MBSE MEASA that defines the process steps, systems engineering products, simulation characteristics, experimental design techniques, and analysis methodologies that distinguish each phase of the MBSE MEASA. To ensure consistency with current efforts in the MBSE community, this research uses SysML products to define the "systems engineering products" appropriate for use in the analysis methodology and demonstrates the use of those products to support simulation models.

This research assumes that previously conducted stakeholder analysis establishes a system need. Accordingly, this work focuses on system simulation techniques that assume that the system of interest has, at least broadly, been defined. This research focuses on the conceptual design phase of system development and assumes simulation models conduct system testing. Note that simulation models can also be used earlier in the system life cycle to aid in stakeholder analysis and development of a concept of operations as well as system requirements (the Institute of Electrical and Electronics Engineers (IEEE) hosts an annual requirements engineering conference (http://www.re15.org/) that demonstrates the power and utility of engineering requirements), but that usage is not the focus of this research. This research focus may prompt additional work into development of systems engineering and modeling and simulations techniques earlier in the system life cycle, or the development of engineering methodologies for ill-defined systems, as well as systems of systems.

THIS PAGE INTENTIONALLY LEFT BLANK

II. PRIOR WORK

Before demonstrating the utility of the MBSE MEASA, this chapter formally identifies the purpose of systems engineering process models in the abstract and reviews the motivation of MBSE, recent developments in MBSE, and important developments in simulation and analysis.

A. SYSTEMS ENGINEERING CONTEXT

The systems engineering process provides guidance for the development of the MBSE MEASA. However, discussion of "the systems engineering process" inevitably transitions to discussion of candidate systems engineering process models, which typically advocate a particular approach to following the more general systems engineering process. This research necessarily follows a similar path, but it is useful to consider the overall goal of a systems engineering process. Per the *Systems Engineering Handbook*, "the SE process has an iterative nature that supports learning and continuous improvement. As the processes unfold, systems engineers uncover the real requirements and the emergent properties of the system. Complexity can lead to unexpected and unpredictable behavior of systems" (SE Handbook Working Group 2011, 8). While this may not provide a comprehensive definition of a systems engineering process, this statement does makes it clear that a systems engineering process should lead to learning, continuous improvement, discovery of requirements, discovery of system properties, and discovery of system behavior.

The *Systems Engineering Handbook's* explanation of a systems engineering process model is a useful starting point for developing an understanding of the systems engineering process. However, the primary objective of this research is development of the MBSE MEASA (which is intended to be implemented within the context of the general systems engineering process); therefore, it is critically important that fundamental characteristics of a systems engineering process be stated. Furthermore, this research requires a review of candidate systems engineering process models within the context of this general characterization of the "systems engineering process" to define how the

newly developed MBSE MEASA should be implemented within a systems engineering process model.

MIL-STD-499A and MIL-STD-499B provide a straightforward definition of the systems engineering process. (The author recognizes that MIL-STD-499A was superseded by MIL-STD-499B on 24 August 1993; however, several of the definitions provided in Revision A are considered clearer and more concise than the definitions provided in Revision B.) Using the definitions of Engineering Management and Systems Engineering Process from MIL-STD-499A, coupled with the definition of Systems Engineering and Systems Engineering Process from MIL-STD-499B, a systems engineering process can be succinctly defined as: "a comprehensive, iterative, problem-solving process (defined by a logical sequence of activities and decisions) that generates information for decision makers by transforming an operational need into a description of system performance parameters and a preferred system configurations." Decomposition of that definition identifies four characteristics of the general systems engineering process.

- The process must be comprehensive. It must not focus on individual aspects of the system and instead should consider the system as an integrated whole.
- The process must be iterative. It must consider an initially stated operational need and evaluate system configurations against that need. The process must simultaneously scope the operational capabilities of the system such that the process can be repeated for a more focused operational need.
- The process must define a logical sequence of activities and decisions. As noted, the process must be iterative, but there is necessarily an element of sequence. The process must explicitly define the ordering and characteristics of each event in the process. Ambiguity must be kept to a minimum in order to clearly delineate each event and clearly define the achievements that trigger the transition between events.
- The process transforms operational needs into descriptions of the system in the form of system performance parameters as well as preferred system configurations. This is perhaps the most important characteristic of a quality systems engineering process. In short, the objective of any systems engineering process is to ensure that the decisions that lead to

recommendation of a system configuration can be directly linked to a clearly defined operational need.

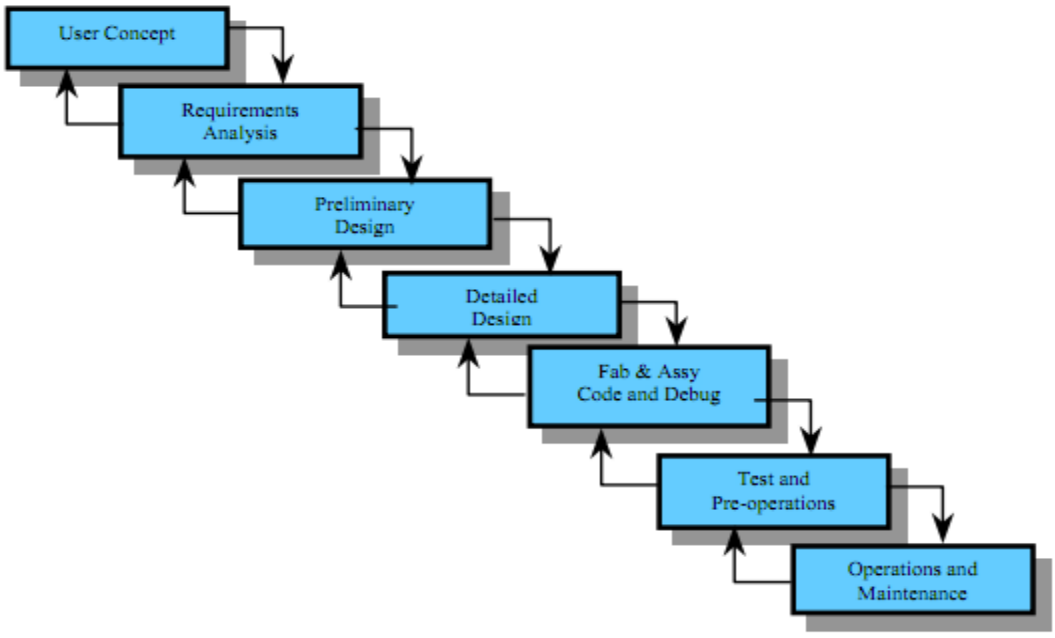
These four process characteristics have been synthesized by the author from each of the sources presented previously and are considered to be fundamental to any quality systems engineering process. Accordingly, the author describes several well-known SE process models and shows how MBSE can be integrated within a generic SE process model.

Stating these general characteristics of the systems engineering process facilitates comparisons between four distinct widely used systems engineering process models. Popular systems engineering texts, such as Blanchard and Fabrycky (2010), Sage and Armstrong (2000), and Buede (2009) describe each of these models, suggesting that they provide a reasonable overview of existing models. Figures of each process model are provided for the unfamiliar reader. Note that many versions of each process model exist, and it is possible (and recommended by most texts) to choose a process model that is tailored to specific problems. The figures shown in this Chapter provide domain neutral visualizations of each process model and provide a clear representation of each process model. Assessment of clear, domain neutral representations of each process model facilitates identification of the mechanisms within each model that most clearly address the overall goals of a "systems engineering process."

The waterfall model, developed by Royce (1970), is the oldest systems engineering process model. The waterfall model is a useful starting point due to the extensive history and documentation of the model (Figure 4). The model espouses a set of distinct, sequential steps, beginning with concept definition and requirements analysis. This ensures resources are not wasted early in the process. System design, coding, and testing immediately follow the requirements analysis. Finally, the system is fielded, operated, and maintained (suggesting that the model satisfies both process characteristics 1 and 4). This sequential, distinct process allows for segmentation of tasks and easy identification of deliverables required to progress from one step to the next (satisfying process characteristic 3). However, very few system productions can follow a rigid, linear set of processes. Development processes are interdependent and cannot be viewed as a

linear series of events. Such a representation is an oversimplification of the complexities associated with system design (Forsberg, Mooz, and Cotterman 2005). More importantly, there exist no mechanisms for the introduction of new system capabilities. The model is inflexible and does not allow for large design changes. It is evident that any attempt to introduce design changes would prove extremely difficult. If forced to integrate a new system capability, the entire process would likely need to be completed and restarted with the change defined as part of the system from the beginning of the process. Note that sequentially linking several waterfall models results in iteration (somewhat satisfying process characteristic 2), but the model is certainly not tailored to redefine the operational need for each iteration. When implemented properly the process model certainly can be used to enable successful system development, but more recent process models have been developed that explicitly address some of the shortcomings of the waterfall model.

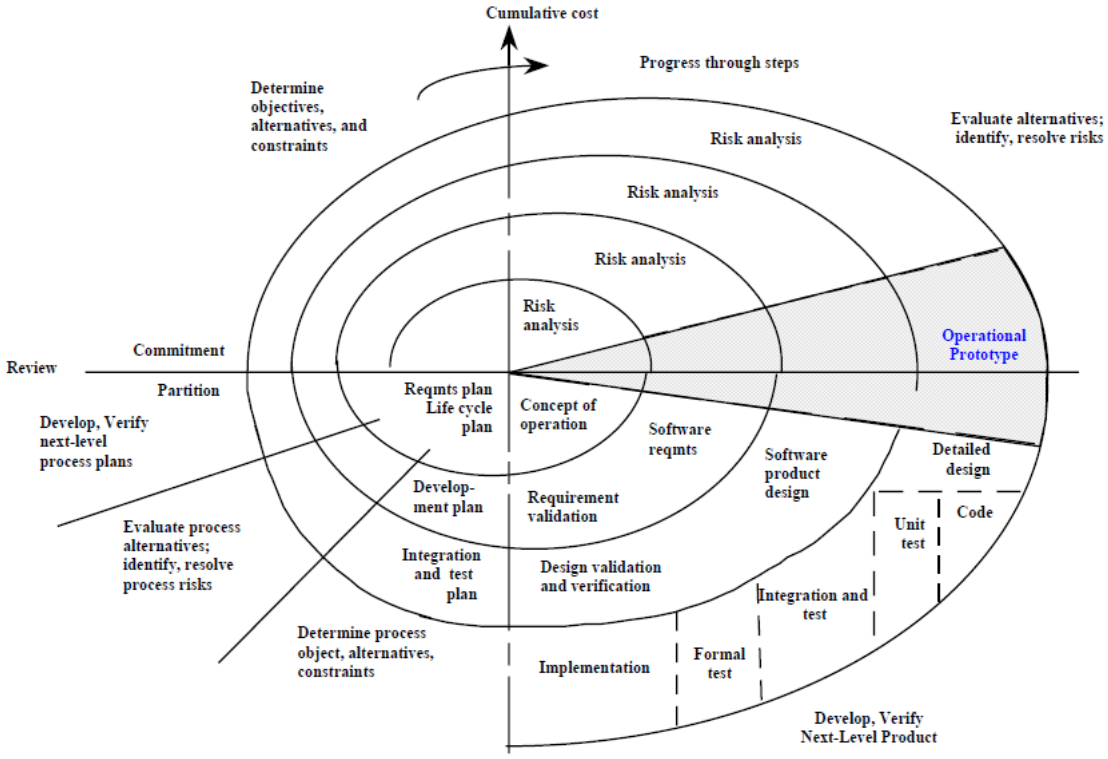
Figure 4 Waterfall Model



Source: Florida Department of Transportation. 2003. *A Process Review and Appraisal of the Systems Engineering Capability for the Florida Department of Transportation (FDOT).* Technical Memorandum No. 1. Tallahassee, FL: Florida Department of Transportation.

The spiral model of system development, first introduced in Boehm (1986), provides an interesting contrast to the waterfall model (Figure 5). While it was evident that the introduction of new operational needs and system capabilities into the waterfall model would prove quite difficult, the spiral model assumes that available technologies will change over the system development timeframe, and therefore, the process model is robust to new operational needs and potential system capabilities. The spiral model assumes that new technologies, capabilities, and needs are introduced during the system life cycle, and therefore the system must be fielded incrementally in order to take advantage of the technological growth that occurred during the system development (the process model is explicitly iterative, satisfying process characteristic 2). The spiral model is essentially a sequence of waterfall models, and after each iteration of the model a smaller system or subsystem is developed and new technologies are introduced (satisfying process characteristic 1). This model requires frequent problem redefinition as well as prototype recreation to allow for the introduction of new technologies. Over the course of system development the system is constantly redefined and redesigned. While a set of activities and decisions are presented in the model, there is potential ambiguity regarding the point at which an appropriate level of system development has been reached to trigger a new design cycle (somewhat satisfying process characteristic 3). More importantly, this may result in system delays as well as prevent a final, understandable definition of the system itself (making satisfying process characteristic 4 difficult using the spiral model). This ambiguity regarding system definition is the basis of most criticisms of the spiral. A particularly notable failure was the use of spiral development for the U.S. Army's Future Combat System, as detailed in Ellman (2009). However, as noted in Farr (2011), the spiral model remains particularly useful for large, expensive, complicated systems where technological change is inevitable and the final system form or configurations is difficult to define early in the system life cycle.

Figure 5 Spiral Model

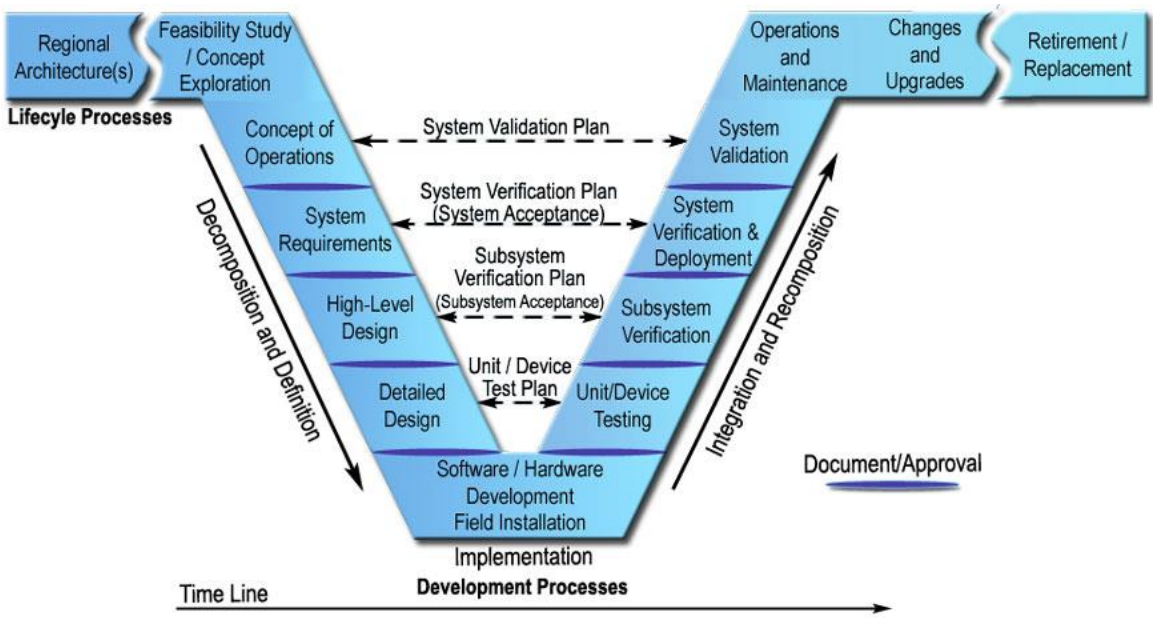


Source: Florida Department of Transportation. 2003. *A Process Review and Appraisal of the Systems Engineering Capability for the Florida Department of Transportation (FDOT).* Technical Memorandum No. 1. Tallahassee, FL: Florida Department of Transportation.

The Vee Model (Figure 6), first developed for systems engineering in the late 1980s, is an attempt to approach systems engineering from both a top down and a bottom up perspective (meaning that it focus both on decomposition of system requirements and integration of system components). The top down (left) portion of the Vee Model is similar to the waterfall model approach; system requirements are identified and decomposed into a particular system configuration. That system is then integrated with new technologies and developing subsystems during the bottom up (right) side of the Vee Model (satisfying process characteristic 1). This clearly identifies a sequence of activities and decisions (satisfying process characteristic 3) and, provided that the completion of a system phase is defined by a milestone, the process may be iterated. This somewhat satisfies process characteristic 2, but, as with the spiral model, each milestone must be

properly defined by a set of requirements that must be met in order to continue system development. Many texts, most notably Foorsberg, Mooz, and Cotterman (2005), advocate system development through a linked set of Vee Models and provide guidelines regarding management of large scale projects and definition of appropriate milestones using the Vee Model. However, the model itself does not provide guidance regarding the development of these requirements or milestones. Perhaps better than the waterfall model or the spiral mode, the Vee Model does build toward a preferred system configuration, satisfying process characteristic 4.

Figure 6 Vee Model

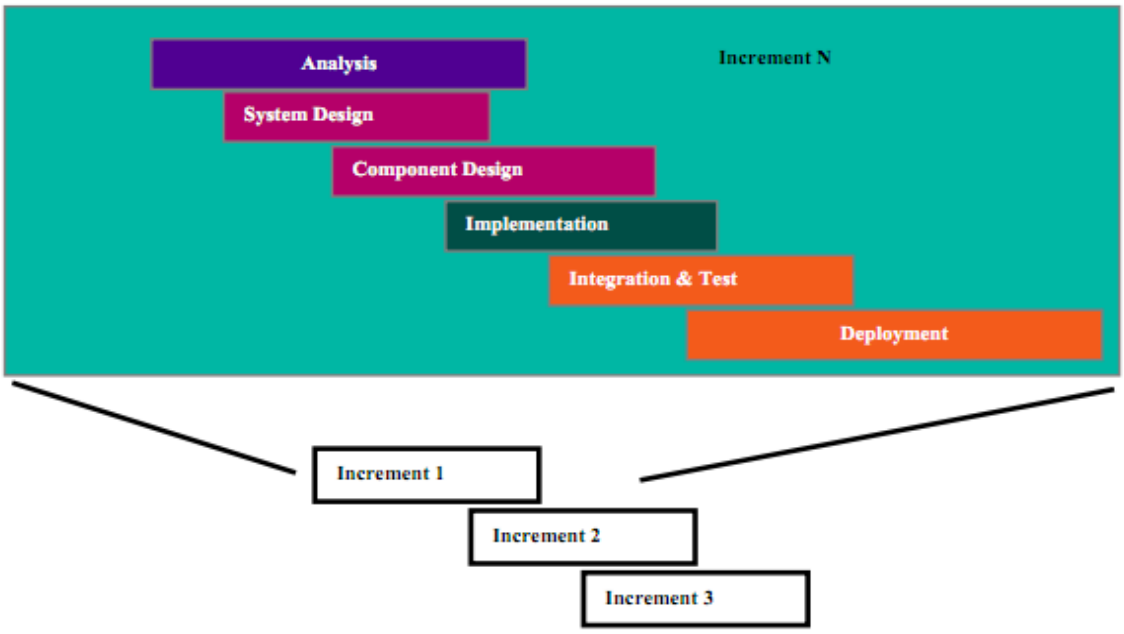


Source: FHWA Operations. 2013. "Systems Engineering for ITS Handbook - Section 3 What Is Systems Engineering?" Dec 9. <http://ops.fhwa.dot.gov/publications/seitsguide /section3.htm>.

The incremental model, first documented in the mid-1970s, is the final systems engineering process model of interest. The incremental model (Figure 7) uses the simplicity and well defined structure of the waterfall model, but rather than aggregating all system functions, divides each functional element of the system into an increment and develops each increment distinctly (implicitly satisfying process characteristic 1). Effectually, each system functional element (or subsystem) is developed using a waterfall

model, defining a set of system development activities (satisfying process characteristic 3). This shifts the focus from a final, large deliverable to multiple, smaller deliverables. The overall system functionality can be broken down into distinct development efforts. The model stresses that each system functional element is a cohesive part of the larger system (which suggests that satisfying process characteristics 4 may be difficult, but that the challenge is not ignored by the model). As a result, parallel development of subsystem components is possible, as well as parallel development of subsystem alternatives. New system capabilities can be introduced to each functional element throughout the system development timeframe because each functional element is being developed individually, and the larger system is not impacted by smaller changes. Furthermore, the incremental model implicitly creates product development cycles that are more independent than spiral model cycles but allow for integration of new capabilities without impacting the overall development timeframe (facilitating easy iteration and satisfying process characteristic 2).

Figure 7 Incremental Model



Source: Florida Department of Transportation. 2003. *A Process Review and Appraisal of the Systems Engineering Capability for the Florida Department of Transportation (FDOT).* Technical Memorandum No. 1. Tallahassee, FL: Florida Department of Transportation.

Each of the systems engineering process models is capable of satisfying each of the process characteristics outlined previously if implemented properly. Selection of a particular process model is often domain dependent. For example, the *Systems Engineering Handbook* suggests that the Vee Model is often more popular for project management applications, while the Spiral Model is more population in software engineering applications. While each of the process models defines a slightly different approach and set of activities, considering each of the systems engineering processes models presented in conjunction with the general definition and characteristics of systems engineering process models presented earlier, it is possible to generate a generic systems engineering process that summarizes each systems engineering process model. Recall that MIL-STD-499A and MIL-STD-499B identify the four characteristics of a systems engineering process as: the process must be comprehensive, the process must be iterative, the process must be defined by a logical sequence of activities and decisions, and the process must transform "an operational need into a description of system performance parameters and a preferred system configuration" (Department of Defense 1974, 3). Summarizing each of these systems engineering process models into a generic systems engineering process is necessary. This ensures that the MBSE MEASA is implementable within a more general systems engineering process model. This allows for definition of consistent terminology and defines how a model-based systems engineering analysis methodology integrates with the general systems engineering process (use of a specific process model may be problematic given that differing terminology is used in each systems engineering process model). Accordingly, the following steps are identified as vital to a single iteration of any systems engineering process model (recall that each process model emphasizes the importance of iteration, which may occur between each step as well as at the conclusion of the implementation of the sequence):

1. Problem Definition

i) Stakeholder Analysis
ii) Requirements Identification

2. System Design

i) Functional Architecture Development

ii) Physical Architecture Development
iii) Allocated Architecture Development
iv) Modeling and Simulation

3. System Analysis

i) Assessment of System Designs
ii) Cost Analysis

4. System Implementation

i) System Production
ii) System Deployment
iii) System Operation
iv) System Disposal

Chapter III discusses this generic process in more detail, provides detailed descriptions of each stage, and links the MBSE MEASA to the generic process.

B. MODEL-BASED SYSTEMS ENGINEERING DEFINITION AND REVIEW

As with most subjects within Systems Engineering, a clear, concise definition serves as a useful starting point for understanding the MBSE. Fortunately, INCOSE's Systems Engineering Vision 2020 defines MBSE as, "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (Technical Operations, INCOSE 2007, 15). When viewed in the context of the systems engineering process outlined previously, the goal of "supporting system requirements, design, analysis, verification, and validation" can be realized through adherence to an appropriate systems engineering process model. However, the definition provided by INCOSE also stresses that each of those activities be supported by "the formalized application of modeling." Accordingly, this research develops an MBSE MEASA that explicitly states how the modeling process supports each activity in the systems engineering process (system requirements, design, analysis, verification, and validation).

1. Introduction and MBSE Progression

While the INCOSE definition of MBSE is a useful starting point, it may remain unclear why MBSE is a useful expansion of systems engineering. The intended benefits of MBSE, presented at the INCOSE 2007 Symposium, provide clarification. Friedenthal, Griego, and Sampson (2007) state that MBSE results in the following benefits:

1. Improved communications among the development stakeholders
2. Increased ability to manage system complexity by enabling a system model to be viewed from multiple perspectives, and to analyze the impact of changes
3. Improved product quality by providing an unambiguous and precise model of the system that can be evaluated for consistency, correctness, and completeness
4. Enhanced knowledge capture and reuse of the information by capturing information in more standardized ways and leveraging built in abstraction mechanisms inherent in model driven approaches. This in-turn can result in reduced cycle time and lower maintenance costs to modify the design
5. Improved ability to teach and learn systems engineering fundamentals by providing a clear and unambiguous representation of the concepts (Friedenthal, Griego, and Sampson 2007, 7)

These intended benefits are adapted into criteria that can assess the ability of a methodology to realize the intended benefits of MBSE. The assessment of the fitness of the MBSE MEASA based on those criteria and the systems engineering process characteristics outlined in the previous section.

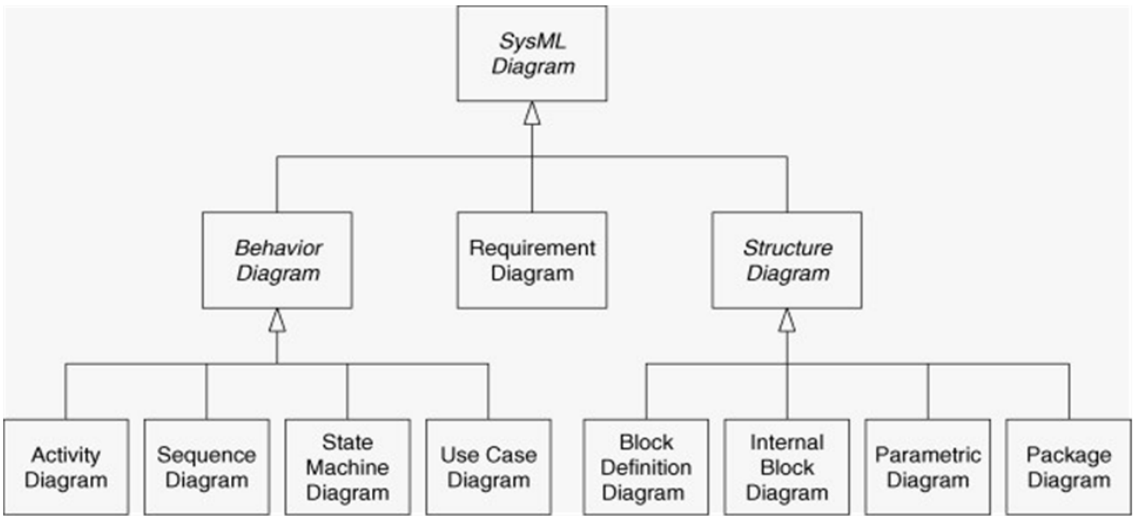
Estefan (2008) provides a comprehensive overview of many existing MBSE methodologies, and therefore serves as an excellent starting point for reviewing several existing methodologies. Before reviewing each of those methodologies in detail, it is useful to review the most well-known MBSE enabler, the Object Management Group's Unified Modeling Language (UML) and Systems Modeling Language (SysML). Discussion of SysML provides a nice transition from a discussion of general systems engineering process models to MBSE methodologies. SysML provides a framework for capturing the maximum possible information about a system in a model-based structure rather than specifying mechanisms for system development decisions. Furthermore, because SysML enables the system model to be viewed from multiple perspectives in a standardized form, use of SysML products as the starting point of the MBSE MEASA

ensures that the methodology is grounded in an enabler that was developed specifically to realize the intended benefits of MBSE. Specifically, this ensures that the architecture domain portion of this research aligns with the most broadly used MBSE architecting approach.

2. SysML Overview

Friedenthal, Moore, and Steiner (2009) provide a clear definition of SysML, stating, "SysML is a general-purpose graphical modeling language that supports the analysis, specification, design, verification, and validation for complex systems." This definition of SysML aligns closely with the previously presented definition of MBSE. SysML attempts to satisfy each of these stated goals through a formal definition of various diagrams, specifically a requirement diagram, an activity diagram, a sequence diagram, a state machine diagram, a use case diagram, a block definition diagram, an internal block diagram, a parametric diagram, and a package diagram. Figure 8 is a taxonomy diagram that more clearly establishes the intended linkage between these diagrams.

Figure 8 SysML Diagram Taxonomy



Source: Friedenthal, Sanford., Alan Moore, and Rick Steiner. 2009. A Practical Guide to SysML The Systems Modeling Language. San Francisco, CA: Morgan Kaufmann Publishers.

Any system developed following a SysML framework should be able to avoid the development of products or system components that do not support the overall system concept due to the hierarchical structure of the taxonomy. This discussion examines each diagram in detail later, but this high level overview of the SysML diagram taxonomy makes it immediately clear that SysML is used "to capture the system modeling information as part of an MBSE approach without imposing a specific method on how this is performed," (Friedenthal, Moore, and Steiner 2009, 31). This statement makes it easier to contrast the overall goals of SysML with the goals of the MBSE MEASA being developed by this research as well as the various existing MBSE methodologies. SysML supports various MBSE development methodologies, but does not specify any preferred method. By specifying a standard set of products (as shown in Figure 8), utilization of SysML aids in the realization of several of the intended benefits of MBSE (specifically: improving communication between stakeholders; defining a model of the system that can be evaluated for consistency, correctness, and completeness; and standardizing information capture to facilitate reuse of information). Friedenthal, Moore, and Steiner (2009) state that SysML can be used to support various system development approaches, such structured analysis use case-driven approaches, or object-oriented approaches. SysML diagrams support the MBSE MEASA due to the popularity of SysML within the MBSE community and the benefits of SysML outlined previously. A detailed discussion of each SysML Diagram is included for the unfamiliar reader. Chapter III presents examples of each type of SysML diagram in the context of a U.S. Navy mine countermeasures operation.

a. SysML Requirement Diagram

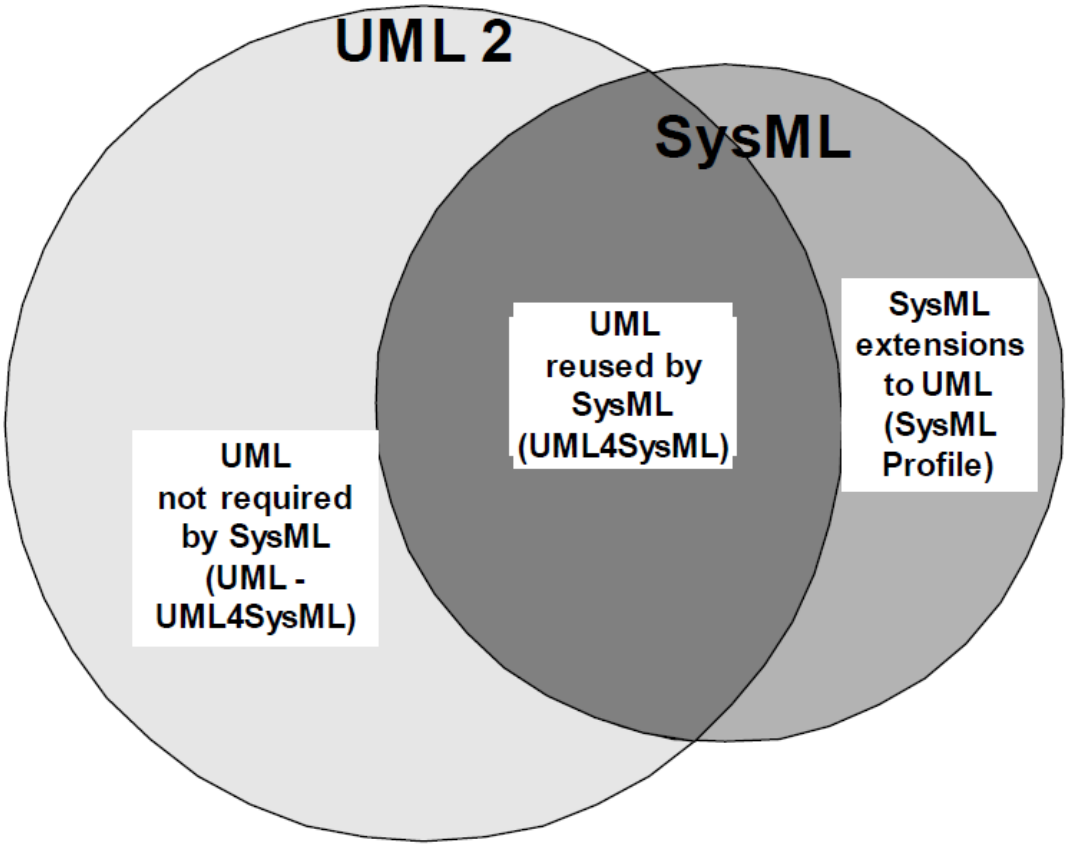
Discussion of the Requirement Diagram Review is a logical starting point for review of SysML diagrams. In an effort to improve communication between systems engineers and the other participants in the systems engineering process, INCOSE released "UML for Systems Engineering," a request for proposal that defined the need for a systems modeling language. The proposal emphasized that a language similar to UML, which is the standard modeling language for software engineering, could not be directly translated to support systems engineering projects. The proposal recommended that UML

be customized for systems engineering to "support the analysis, specification, design, and verification of complex systems by:

1. Capturing the systems information in a precise and efficient manner that enables it to be integrated and reused in a wider context
2. Analyzing and evaluating the system being specified, to identify and resolve system requirements and design issues, and to support trade-offs
3. Communicating systems information correctly and consistently among various stakeholders and participants" (Object Management Group 2003, 1)

SysML was developed using UML as a basis; Figure 9 shows the relationship between SysML and UML.

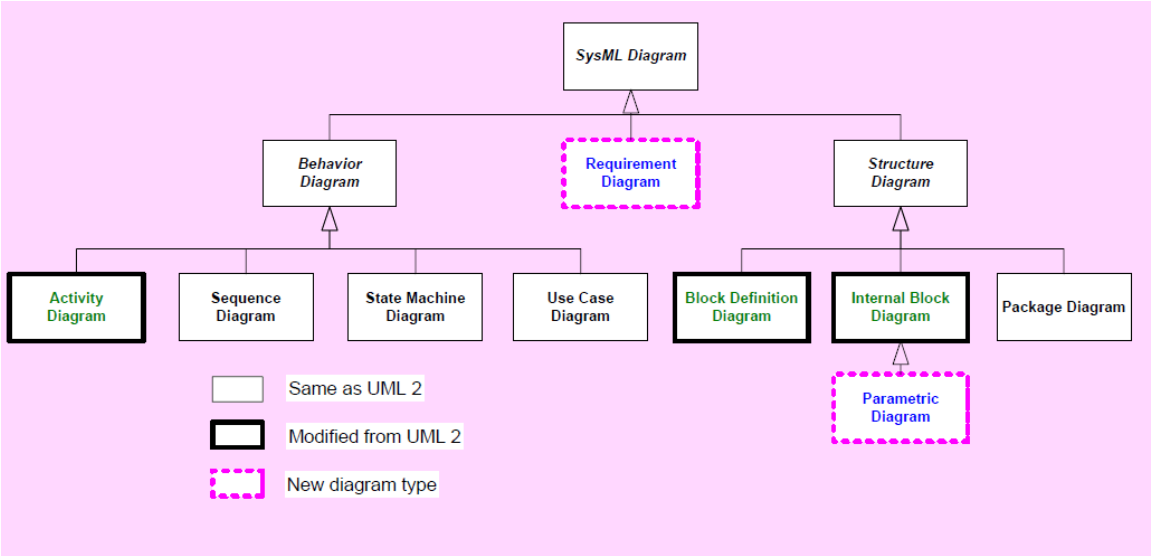
Figure 9 Relationship Between SysML and UML



Source: Object Management Group. 2012. OMG Systems Modeling Language (OMG SysML) Version 1.3. OMG document number ptc/2012-04-07.

UML was extended by SysML to support modeling of general systems, rather than only modeling of software systems. Figure 9 shows that a subset of UML was reused or modified for use in SysML, while portions that were not needed for systems modeling were excluded. Additionally, new diagrams were developed to capture system information that is not needed for software modeling. Figure 10 shows the specific diagrams that were reused or created for SysML.

Figure 10 SysML Diagram Taxonomy and Relationship to UML



Source: Object Management Group. 2006. OMG Systems Modeling Language (OMG SysML) Tutorial. Presented at the INCOSE 2006 Symposium, Orlando, FL.

Figure 10 demonstrates why a discussion of SysML (in terms of its relationship to UML) should begin with a discussion of the Requirement Diagram. The Requirement Diagram is the most noticeable difference between SysML and UML. While a Requirement Diagram is not included in UML (because the software engineering community understandably focuses development on behaviors and structures), it is a focal point of SysML. The INCOSE definition of systems engineering emphasizes that a major function of systems engineering is "documenting requirements" and the INCOSE definition of a systems engineering process states, "systems engineers uncover real requirements." Note that this does not mean that stakeholders do not provide

requirements, rather it means that systems engineers are tasked with determining whether or not those requirements are "real requirements," or simply things that the stakeholder desires but does not actually require. Given the focus on requirements in the definitions of systems engineering and the systems engineering process, it is unsurprising that the most obvious extension that SysML makes to UML is the specification of a Requirement Diagram.

A Requirement Diagram is used "to graphically depict hierarchies of requirements or to depict an individual requirement and its relationship to other model elements" (Friedenthal, Moore, and Steiner 2009, 538). *Containment, derive,* or *copy* relationships are used to describe requirements to requirements relationships. *Satisfy, verify, refine,* or *trace* relationships are used to relate requirements to other model elements.

b. SysML Activity Diagram

An Activity Diagram "is used to model behavior in terms of the flow of inputs, outputs, and control" (Friedenthal, Moore, and Steiner 2009, 527). It can be used to represent different types of system behaviors, such as control flow or data flow. It is typically used to show sequences of operations and is described in terms of *activities, controls (join, fork, decision, loop), data flows (required or optional), and swim lanes.* Activity Diagrams are similar in purpose and structure to Functional Flow Block Diagrams (FFBD) and Enhanced Functional Flow Block Diagrams (EFFBD), two of the more commonly used systems engineering architecture products. For the unfamiliar reader, Blanchard and Fabrycky (2010) provide an overview of the role of FFBDs and EFFBDs in the systems engineering process and provide a detailed discussion of the alternative graphical approaches (such as Integrated Definition (IDEF) methods, modeling methods, behavior diagram methods, and N-Squared charting methods).

c. SysML Block Definition Diagram

The Block Definition Diagram defines blocks (often the physical elements) of a model. Block Definition Diagrams are particularly useful for defining hierarchical relationships, as well as the structural and behavioral features of each element of the model.

d. SysML Internal Block Diagram

The Internal Block diagram is similar in structure to the Block Definition Diagram, but specifically defines the internal structure of a block (typically a physical element) with a focus on the connections between parts of a block.

e. SysML Sequence Diagram

Sequence Diagrams show interactions. These interactions occur between elements of a block (as defined in the Block Definition and Internal Block Diagrams). Sequence Diagrams are particularly useful for defining sequences of message exchanges or trigger actions between blocks.

f. SysML State Machine Diagram

A State Machine Diagram describes state dependent actions of a block. This allows each block to perform different behaviors, which are mutually exclusive (note that a block may only be in one state at a given time). This ensures that no conflicting responses to events are prescribed. The State Machine Diagram also specifies how transitions between states should occur.

g. SysML Use Case Diagram

The Use Case Diagram describes the behavior of a system, specifically the relationship between a system and actors that impact the operation of that system. Typically Use Case Diagrams represent actors internal to the system of interest (for example, a driver) but depending on the level of abstraction they may also represent the relationship between the system of interest and external actors (for example, a traffic police officer).

h. SysML Parametric Diagram

The Parametric Diagram defines systems of equations that describe the behavior of a block (recall that a block is most often a physical element of a system). The Parametric Diagram constrains properties of blocks and those constraints check for consistency between the physical elements of a system. They can be used as the basis for

the construction of external models or simulations. Parametric Diagrams are most useful during the later stages of system development (when their representation of system properties as defined values is necessary). This research focuses on early stage system development. As such, Parametric Diagrams, which are most useful when systems can be specified by specific constraints, are not a major focus of this research.

i. SysML Package Diagram

Package Diagrams organize SysML blocks. While they do not provide additional functionality, they can aid in organization of stakeholder guidance to ensure proper organization of model elements.

3. Current MBSE Methods and Processes

SysML defines a set of products that can be used to improve communication and cohesion throughout the systems engineering process. Importantly, it does not make any assumptions regarding the implementation of those products or their application within the systems engineering process. As mentioned, MBSE formalizes the application of modeling to support system development. Along those lines, several major companies and organizations have defined MBSE methods and processes, most of which rely on SysML products as enablers of the methods or processes. A useful starting point for identification of the most widely used MBSE methods and processes is the running repository of MBSE methodologies managed by INCOSE. Using INCOSE and Estefan (2008) as a guideline (the Estefan (2008) research was also managed by INCOSE), the most well-known MBSE processes/methods are: IBM Harmony for Systems Engineering, INCOSE Object Oriented Systems Engineering Method, Vitech Model-Based Systems Engineering Methodology, NASA Jet Propulsion Lab State Analysis, Dori Object-Process Methodology, and Weilkiens Systems Modeling Process. Note that each methodology is presented along with the developer (ex: Object Oriented Systems Engineering Method was developed by INCOSE, Object-Process Methodology was developed by Dori). Each of these methods and processes represent an expansion of the general systems engineering process presented earlier. Specifically, they formalize a methodology for integrating a set of models within the general systems engineering

process. A review of each of these methods demonstrates how MBSE is implemented by different organizations and highlights the current gap that the MBSE MEASA addresses.

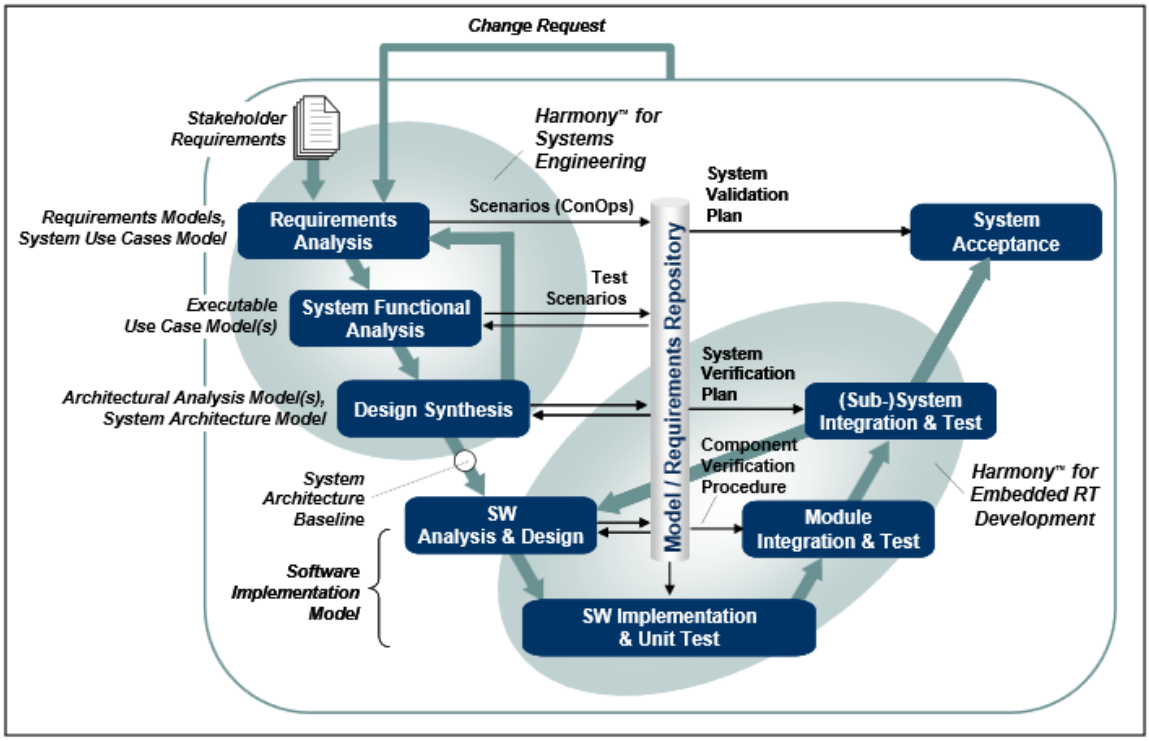
a. IBM Harmony for Systems Engineering

IBM Harmony for Systems Engineering, based largely on IBM's Rational Integrated Systems/Embedded Software Development Process Harmony, supports a model driven development approach to MBSE that is intended to satisfy three major objectives, as presented in Hoffman (2011):

1. Identification of derivation of required system functions
2. Identification of associated system modes and states
3. Allocation of the identified system functions and models/states to a subsystem structure (Hoffman 2011, 4).

The process relies heavily on creation and use of UML/SysML products and is implemented using IBM's Rational Rhapsody tool. Harmony emphasizes that the process develops models that support requirements analysis (through generation of Requirements Models and Use Case Models) as well as design synthesis models (using Architectural Analysis Models and System Architecture Models). The comprehensive Rational Integrated Systems/Embedded Software Development Process Harmony (Figure 11) uses the Vee Model as a basis and provides a guideline for system development.

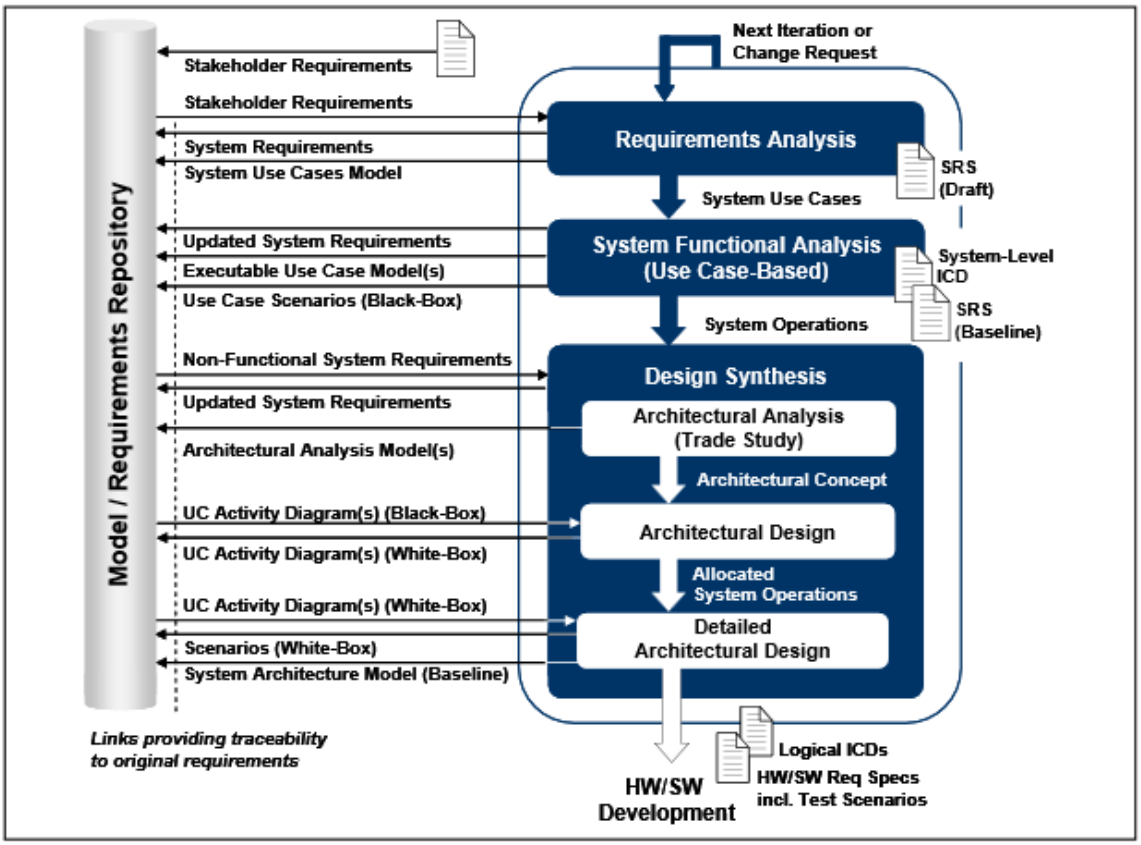
Figure 11 Rational Integrated Systems/Embedded Software Development Process Harmony



Source: Hoffman, Hans-Peter. 2011. Model-Based Systems Engineering with Rational Rhapsody and Rational Harmony for Systems Engineering, Release 3.1.2. Somers, NY: IBM Corporation.

Note that the process includes each of the portions of the general SE process (Problem Definition, System Design, System Analysis, and System Implementation). Harmony describes how UML/SysML products support each segment of the general process (Figure 12).

Figure 12 Linkage of Model Artifacts to Systems Engineering Process Steps



Source: Hoffman, Hans-Peter. 2011. Model-Based Systems Engineering with Rational Rhapsody and Rational Harmony for Systems Engineering, Release 3.1.2. Somers, NY: IBM Corporation.

Note that the title of Figure 12 is "Model-based Systems Engineering" in the original document. The author altered the title for clarification and consistency with other MBSE methodologies. Examination of Figure 12 demonstrates that IBM Harmony for Systems Engineering defines the artifacts/models, as well as the work-flow elements transition from Stakeholder Input to Requirements Analysis to System Functional Analysis to System Architectural Design. Most importantly, Hoffman (2011) describes the overall work-flow as well as a use case example that demonstrates which SysML products are required to support the overall process. Note that the analysis of system performance is addressed through examination of scenarios during the detailed architectural design and relies largely on generation of utility curves for each

performance criterion, not through the use of external simulations. This is intentional; as Fisher (2013) emphasizes that the IBM Rational Rhapsody tool is best utilized to serve as a central design hub to enable stakeholder collaboration and document generation and reporting, all to realize coordinated and correct system architecture and design. This highlights the difference between IBM Harmony for Systems Engineering and the MBSE MEASA. Harmony for Systems Engineering focuses on improving collaboration and communication through definition of coordinated SysML products, while the MBSE MEASA uses those SysML products to support system performance analysis through external simulations.

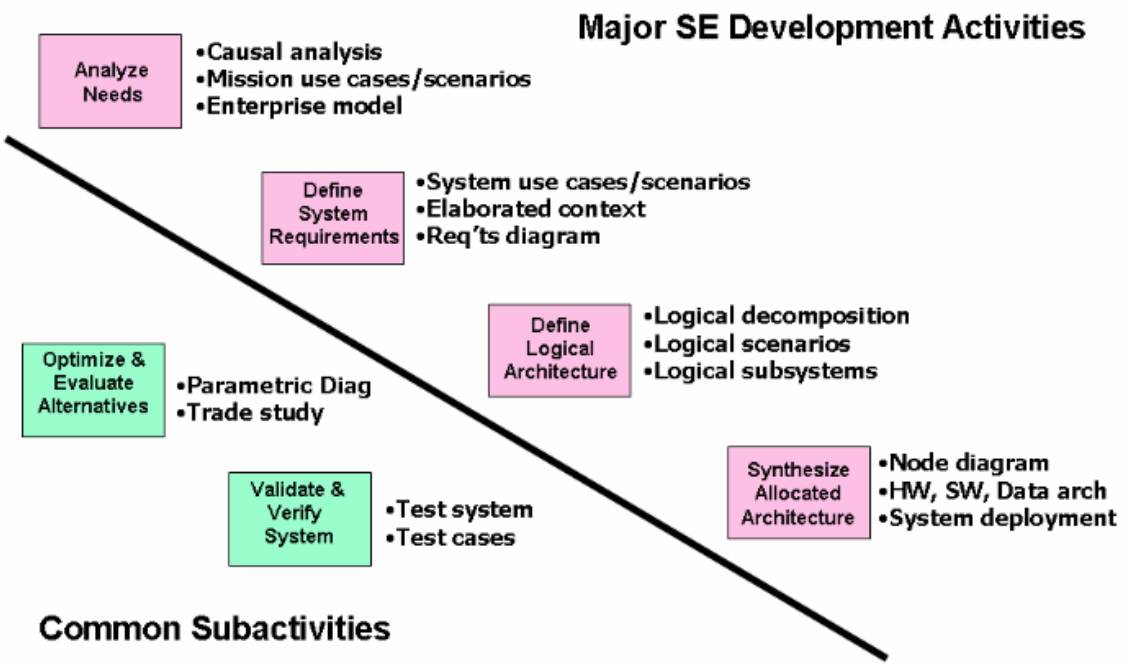
b. INCOSE Object-Oriented Systems Engineering Method

INCOSE Object Oriented Systems Engineering Method (OOSEM) is an alternative methodology that also relies heavily on generation of SysML products. OOSEM is an attempt to integrate traditional systems engineering process models with object-oriented techniques typically used in the software engineering community. Specifically, INCOSE (2011) states that OOSEM defines notation and concepts that:

1. Support capture, analysis and understanding of complex systems specifications and design
2. Improve integration between systems, software, hardware, test, and other engineering disciplines
3. Facilitate system, element, and component level reuse and design evolution (INCOSE 2011, 1)

Like IBM's Harmony for Systems Engineering, OOSEM mimics the traditional systems engineering Vee Model. Note that OOSEM emphasizes that progression through the Vee Model is not a terminating, linear set of processes, but rather should be applied recursively and iteratively (as recommended in the review of the generic Vee Model). Figure 13 provides a visual description of the OOSEM activities.

Figure 13 OOSEM Activities and Modeling Artifacts



Source: Estefan, Jeff A. 2008. Survey of Model-Based Systems Engineering (MBSE) Methodologies, Rev B. Pasadena, CA: California Institute of Technology.

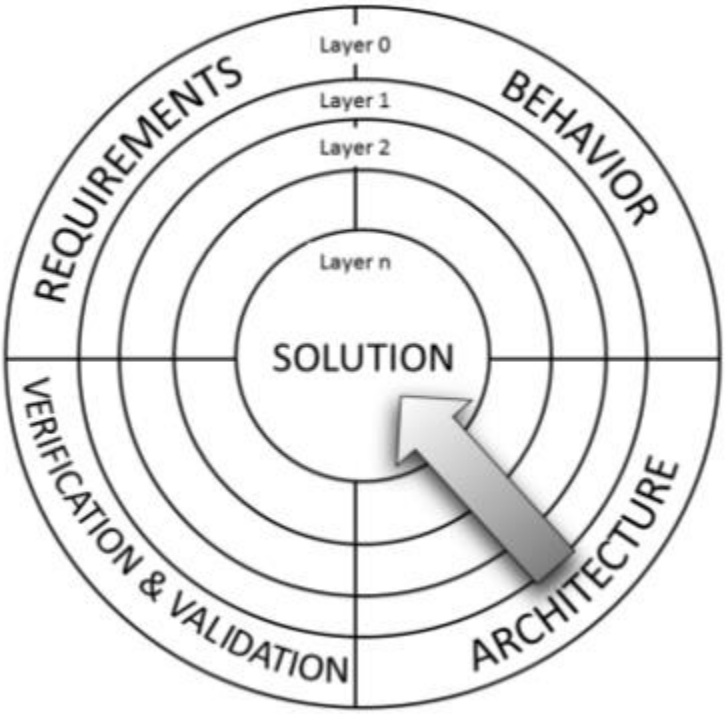
Figure 13 clarifies the approach advocated by OOSEM. The OOSEM appears to mirror the Waterfall Model rather than the Vee Model. Recall that the Vee Model, which includes many of the same activities as the Waterfall Model, emphasizes the relationships between each system development activity and the integration of system components. OOSEM specifies the relationships between activities and the integration of system components. It may help with visualization of OOSEM as a Vee Model-based methodology to "bend" the major SE Development Activities upwards after the Define System Requirements block (and Optimize and Evaluate Alternative and Validate and Verify System should certainly be included on this upwards portion of the Vee). On that subject, the Major SE Development Activities (above the line) make it clear that the OOSEM provides a roadmap for system development, beginning with a needs analysis and concluding with a synthesized allocated architecture (which should ensure that all physical system elements satisfy defined system functions). Finally, OOSEM regards system testing and analysis as processes that are distinct from major development

activities (see their classification as “Common Subactivities” below the line). Because OOSEM is intended to be realized through creation of SysML products, it advocates analysis of system performance through use of parametric diagrams, which are used to optimize individual system architectures using weighting factors and value measures (largely similar to IBM’s Harmony for Systems Engineering). External modeling and simulation is not described as a part of OOSEM. Friedenthal, Moore, and Steiner (2009) provide a comprehensive overview of using SysML products to enable OOSEM and acknowledge that external models and simulation may be valuable in examining system performance, but no formal linkage between SysML products or OOSEM with external models and simulations is established.

c. Vitech Model-Based Systems Engineering Methodology

Vitech’s Model-Based Systems Engineering Methodology is based on the tenant that there are four major domains of the systems engineering process, “requirements, functional behavior, architecture/synthesis, and design validation and verification” (Vitech Corporation 2011, 66). The methodology further advocates solving each domain at increasing layers of granularity, progressing toward realization of a complete system. The methodology refers to this progression as “onion layers,” Figure 14 illustrates the approach.

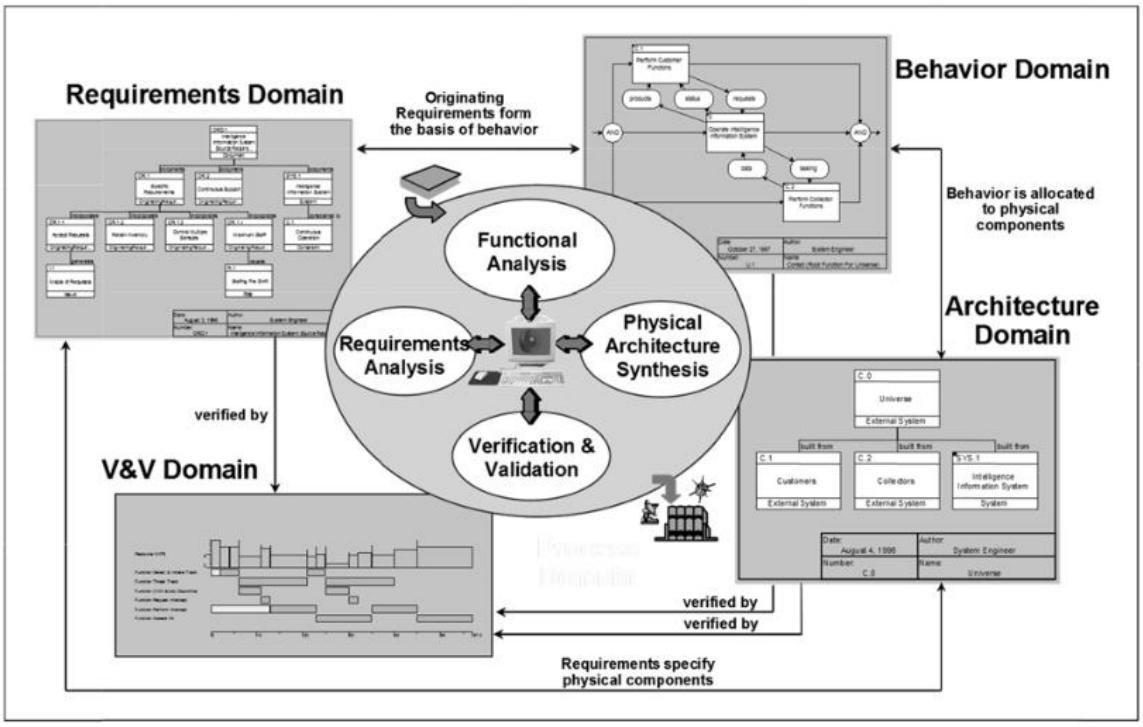
Figure 14 Onion Layers for Vitech's Model-Based Systems Engineering Methodology



Source: Vitech Corporation. 2011. *A Primer for Model-Based Systems Engineering*. Blacksburg, VA: Vitech Corporation.

Vitech's Model-Based Systems Engineering Methodology (Figure 15) specifies the sequencing within each layer.

Figure 15 Systems Engineering Activities for Vitech's Model-Based Systems Engineering Methodology



Source: Vitech Corporation. 2010. *CORE 7 System Definition Guide.* Blacksburg, VA: Vitech Corporation.

The progression of the systems engineering activities moves clockwise, beginning with the Requirements Domain (note that a slightly revised version of the figure has been developed by Vitech since 2010, but the author feels that the revised versions, while aesthetically superior, actually provide less information). The methodology defines products within the Requirements Domain, which specify the products in the Behavior Domain (typically system functions), which generate products in the Architecture Domain (typically physical system alternatives), which are assessed in the Verification and Validation Domain. Note that Vitech's use of the term Architecture Domain differs from the use of the term earlier in this dissertation. As used by Vitech, Architecture Domain refers solely to physical system alternatives, and while it is linked to functions and system behaviors, it does not include those products (which are included as part of the Architecture Domain as the term in used in Chapter I). As with both IBM's Harmony

for Systems Engineering and INCOSE's OOSEM, this progression aligns nicely with the general systems engineering process. Vitech departs slightly from the IBM and INCOSE defined methodology in the verification and validation domain. Rather than relying on SysML parametric diagrams to assess system performance, Vitech advocates use of CORESim, a dynamic verification simulation that checks system architecture models for logical consistency and physical model consistency that is executable within Vitech's proprietary software program, CORE. Note that Vitech's entire methodology is intended to be supported within CORE, similar to the support that IBM offers for Harmony for Systems Engineering with the Rhapsody tool. The CORE tool can support creation of SysML diagrams as well as more traditional systems engineering architecture artifacts. While the implementation of CORESim is different from the use of Parametric Diagrams, there are very few practical differences. CORESim interprets system behavior, as defined previously in Functional Flow Block Diagrams (which are nearly equivalent to generic versions of SysML Activity Diagrams). Parametric Diagrams interpret system behaviors, as defined previously in Activity Diagrams. System performance characteristics are defined in both cases using probabilistic functions and weighting criteria. Both approaches establish traceability between previously established system architecture products and provide a mechanism for verifying the integrity of those models. Such an approach is extremely valuable and powerful for ensuring consistency, completeness, and correctness of architecture models. However, none of the approaches provide a mechanism for efficiently and comprehensively analyzing the impact that alterations to system configurations, system operating procedures, or external environment have on system performance.

d. NASA Jet Propulsion Lab State Analysis

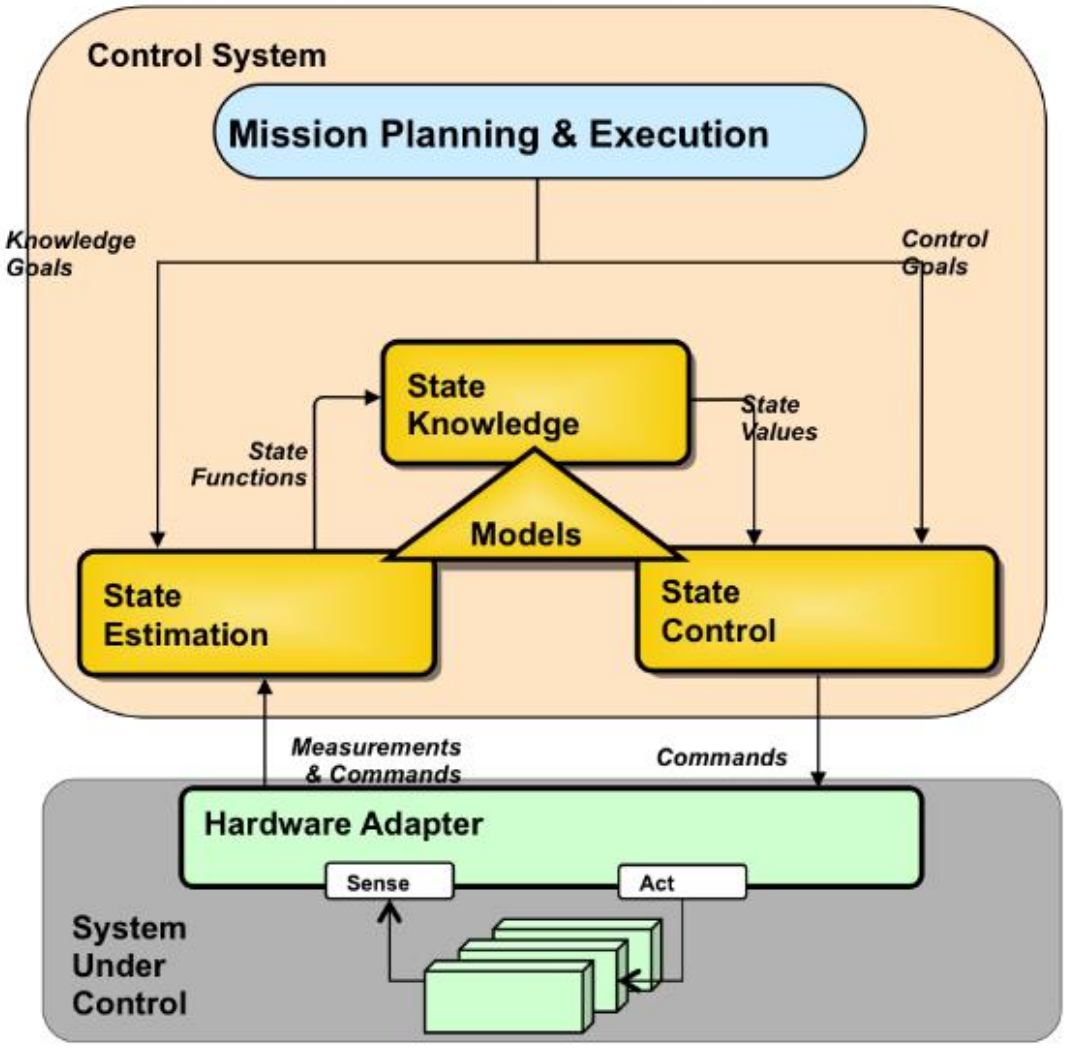
The Jet Propulsion Lab (JPL) State Analysis MBSE methodology is a departure from the previously presented MBSE methodologies. State Analysis is an attempt to integrate both model-based architectures and state based architectures. The approach is drastically different from the architecture view based approach advocated by the IBM, INCOSE, and Vitech methodologies and instead resembles a control systems approach to MBSE. As defined in Wagner et al. (2012), the State Analysis methodology is intended

to produce a control system architecture, rather than a physical or functional system architecture, by:

1. Discovering, characterizing, representing, and documenting the states of a system
2. Modeling the behavior of state variables and relationships among them, including information about hardware interfaces and operation;
3. Capturing the mission objectives in detailed scenarios motivated by operator intent (Wagner et al. 2012, 3)

The State Analysis methodology is initiated by definition of a physical system and subsequently focuses on modeling the potential states (or momentary system conditions) of that system and the relationships between those states. Control objectives are imposed as mathematical formulas that govern system behavior. The approach does use UML representations (with particular emphasis given to State Chart Diagrams, but also allows for the creation of alternative diagrams, such as Elaboration Diagrams). The State Analysis approach delineates between the system of interest and the control system that governs behavior (this delineation is often quite complex, but may be as simple as the difference between hardware and software). Figure 16 provides a visual representation of this separation:

Figure 16 State Based Control Architecture



Source: Wagner, David A., Matthew B. Bennett, Robert Karban, Nicolas Rouquette, Steven Jenkins, Michel Ingham. 2012. "An Ontology for State Analysis: Formalizing the Mapping to SysML." *Aerospace Conference, 2012 IEEE,* 1–16.

This distinction between the "Control System" and the "System Under Control" improves communication between physical engineers and software engineers by bridging the gap that arises due to differing requirements for each set of engineers. Utilization of JPL State Analysis provides a formal process for developing models of both physical systems, software systems, and the interfaces between them. JPL State Analysis focuses on ensuring that any developed software requirements are tied to previously developed system requirements, thereby eliminating potential gaps or conflicts between the

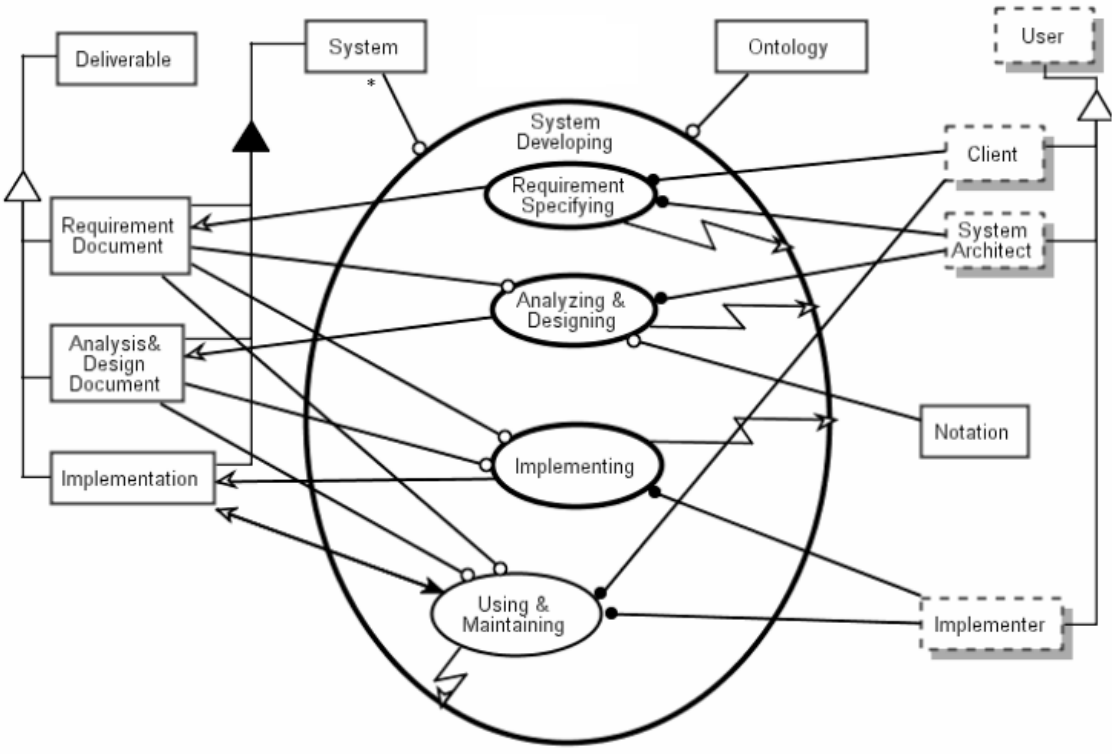
hardware and software domains. This is certainly a tremendously powerful methodology for early stage system development, but focuses primarily on the interactions between hardware and software and is therefore most applicable for software focused systems. The MBSE MEASA presented in this research intends to prescribe a mechanism for developing and analyzing performance models that focus on the interactions between system hardware components as well as the interactions between a system and its physical environment. Accordingly, comparisons between the two methodologies may not be appropriate. Rather, the two methodologies could be applied concurrently during the system design phase, where the MBSE MEASA focuses on system operational performance and JPL State Analysis ensures compatibility between hardware and software requirements.

e. Dori Object-Process Methodology

Object-Process Methodology, developed and refined by Dov Dori and first presented in Dori (2002), is a systems engineering approach that is intended to be domain independent and enables system architecture development and design, primarily focusing on information exchanges between systems. Object-Process Methodology represents systems of interest through both graphics (termed Object-Process Diagrams) and text descriptions (through use of Object-Process Methodology's Object Process Language). Object-Process Methodology is certainly more similar to JPL State Analysis than the IBM, INCOSE, or Vitech methodologies. The major departure from the methodologies presented earlier (which can be viewed as more object oriented methodologies) is that Object-Process Methodology delineates between physical systems (termed "objects" in Object-Process Methodology) and processes as two distinct classes of things that are considered the fundamental basis for any model (not dissimilar to the separation between the System Under Control and the Control System in JPL State Analysis). Object-Process Methodology emphasizes that objects are in different states at different times, and that changes in states are initiated by processes. The methodology focuses on definition of these objects, states, and the processes that initiate changes between states. Dori (2002) formally defines objects as things that exist or may exist; states as situations in which an object may exist; and processes as patterns of change that transform objects by changing

their states. Object-Process Methodology follows a roadmap similar to the general systems engineering process outlined earlier. Dori, Reinhartz-Berger, and Sturm (2003) provide a visualization of the system development processes that occur within Object-Process Methodology implementations (Figure 17).

Figure 17 Object-Process Methodology Progression



Source: Dori, Dov, Iris Reinhartz-Berger, and Arnon Sturm. 2003. "Developing Complex Systems with Object-Process Methodology using OPCAT." Industrial Presentation in Proceedings of the 22nd International Conference on Conceptual Modeling, Chicago, IL.

Figure 17 defines the system development steps for Object-Process Methodology from the top-down. The methodology defines procedures for Requirement Specifying, Analyzing and Designing, Implementing, and Using and Maintaining. The methodology notes that each process can "invoke restarting of the entire development process, which potentially enables the introduction of changes to the requirements, analysis, design, and implementation of the system" (Dori, Reinhartz-Berger, and Sturm 2003, 6). In this way, the methodology allows for iteration not only of the entire process, but of individual steps

of the process. The Analyzing and Designing stage is of particular interest to this research. The stage is initiated by pulling a Requirements Document from the Requirements Specifying stage to enable development of system dynamics and system control structure models (in this way, it is again not dissimilar from the focus of JPL State Analysis). These models are used to identify discrepancies, inconsistencies, and deviations in system behaviors resulting from poor definition of system object and process specification. While the methodology enables rapid examination of analysis of proper linkages between software and hardware systems (much like the JPL State Analysis methodology) the ability to use Object-Process Methodology architecture products to develop detailed external performance models is limited. The methodology does provide a useful extension of JPL State Analysis by explicitly specifying objects and processes that are internal or external to the system of interest (delineating between the system and the external environment) but due to the intended implementation of Object-Process Methodology, it is poorly suited for utilization as a mechanism for conducting detailed performance analysis of large scale, complex systems. It should be noted that, as with JPL State Analysis, Object-Process Methodology could be applied concurrently with the MBSE MEASA developed in this dissertation, as the two approaches examine system performance from different perspectives.

f. Weilkiens Systems Modeling Process

A recent MBSE modeling process specifically focused on utilization of SysML/UML products, presented in Weilkiens (2008), is the Systems Modeling (SYSMOD) Process. SYSMOD presents an approach to definition of system requirements, system functional architecture, and system physical architecture. The SYSMOD process is comprised of a defined set of activities: Identify stakeholders, elicit requirements, define system context, analyze requirements through use cases, define domain model, and define functional, physical, and logical system architecture. The process relies heavily on the use of SysML products (the primary developer of SYSMOD has written several SysML specifications). The mechanisms for conducting stakeholder identification, requirements elicitations, and system context definition are similar to the INCOSE OOSEM approach and result in the production of SysML diagrams. Use Case

Diagrams are the primary mechanism for assessing the quality of system requirements, and accordingly the system performance analysis focuses on system controls (the importance of which was emphasized in the discussion of JPL State Analysis and Object-Process Methodologies) as well as the flow of objects within the system. SYSMOD expands the utility of the competing methodologies by explicitly defining roles for each individual involved in system development (roles include: Administrator, Domain Expert, Process Designer, Requirements Engineer, Systems Analyst, Systems Architect, Systems Engineer, and Systems Tester). Each task within the system development process is assigned to one or more roles, thereby ensuring complete definition of each task. This addition ensures that the SYSMOD process is perhaps the most comprehensive MBSE methodology from a system management perspective. However, it still relies solely on the use of organic SysML products to assess system performance.

As with each of the methodologies outlined previously, SYSMOD effectively establishes that, given a set of functions (based on a set of requirements) a system must be capable of performing in a given set of scenarios (typically represented through use case diagrams). However, no methodology prescribes a mechanism for using existing products to completely define how external system performance models should be built, allowing examination of these use cases in greater detail. The number of system variables (in terms of physical system configurations, system component interactions, system operating procedures, system-environment interactions, etc.) that are examined and assessed using Use Case and Parametric Diagrams is limited. These types of external performance models are essential to examining system performance in detail. The MBSE MEASA developed in this research fills that gap.

As an additional point of emphasis, note that each of these methodologies recognizes that system performance must be analyzed to ensure that proper system requirements are established. More importantly, note that each of the applications relies on some form of mathematical modeling and value function assignment (either through Package Diagrams, CORESim, or Object-Process/State Analysis) to evaluate system performance. The MBSE MEASA extends that approach by formalizing a method for utilizing SysML diagrams to define inputs and outputs to external system performance

models. The MBSE MEASA is not a comprehensive alternative to these existing MBSE methodologies. Rather, by leveraging existing approaches for constructing external simulations and conducting and analyzing large-scale simulation experiments, the MBSE MEASA is a mechanism for expanded performance analysis beyond mathematical models and value function assignment.

4. Recent MBSE Advances

MBSE research has taken on multiple forms, and substantial development has occurred in the last ten years to formalize the various aspects of MBSE. Recent work in MBSE and simulation relevant to this research can be classified into four general areas, MBSE focused system architecting, MBSE-related system analysis, linkage of SysML to simulation, and design and analysis of large scale simulation experiments. Recent advances in system architecture development, specifically the use of SysML products (the utility of which is often evaluated through presentation of case studies and analysis of past projects), must be examined to ensure that a comprehensive definition of how SysML products should enable development of external models and simulations has not been developed. The systems engineering community has focused substantial effort into analysis of SysML utility, but the MBSE MEASA provides a unique formalization of how those products should be used to support development of external models and simulations. Similarly, there has been substantial research, particularly in the area of Engineered Resilient Systems, into the use of models and simulations to enable exploration of large trade spaces. It is necessary to review these advances to reinforce that the MBSE MEASA is being developed in support of areas of emphasis for the larger systems engineering community and that it expands the body of knowledge associated with not only model-based engineering approaches but also model-based system analysis approaches.

a. MBSE Architecture and SysML Development

While each of the MBSE methodologies presented earlier advocated a different theoretical approach and framework to system architecture development, the one common thread was the use of SysML products as the primary enabler to the

methodology (in the case of JPL State Analysis and Object-Process Methodology, SysML was not used as the primary enabler but the popularity of SysML was recognized and procedures for interfacing with SysML were established to facilitate communication between users of those methodologies and the larger MBSE community). Accordingly, it is useful to review recent research into architecture development and SysML use within the context of MBSE.

Per the generic systems engineering process outlined earlier, architecture development typically initiates with definition of a functional architecture. A complete functional architecture translates defined system requirements into defined activities that the system must perform to satisfy those requirements. A review of the importance of functional models to an MBSE approach was demonstrated in Carson and Sheeley (2013), who emphasize that a properly constructed functional architecture serves as a bridge between the problem space (which is primarily defined through requirements analysis) and the solutions space (which is primarily defined by system synthesis models). Through presentation of various examples across a broad range of systems, Carson and Sheeley demonstrate that a poorly defined functional architecture results in issues in the problem space (particularly that system boundaries may be improperly or ill-defined) as well as in the solution space (particularly that systems may exhibit less than ideal performance because they are not developed with emphasis on satisfaction of well-defined functions). This demonstration of the importance of functional architecture development to MBSE focused development is integral to the construction of the MSBE MEASA. Russell (2012) presents similar findings, demonstrating that architecture development in support of MBSE enables understanding of complex interactions and supports decision making by establishing a clear linkage between requirements, metrics, processes, and standards to system design elements. Specifically, it is vitally important to develop a clear functional architecture that defines exactly what a system must do to ensure proper system boundary development and proper operational performance model development.

Summers, Eckert, and Goel (2013) and Wu, Ciavola, and Gershenson (2013), survey various functional modeling techniques and develop criteria for assessing those

approaches. This emphasizes the importance of functional architecture development early in the system life cycle. Summers, Eckert, and Goel emphasize that these criteria may differ depending on the type of system being considered (as an example, functional modeling for reverse engineered systems differs from functional modeling for novel products). Kenley, Dannenhoffer, Wood, and DeLaurenitis (2014) demonstrate that UML products can capture the functionality of a large scale system of systems to support communication and subsequent model development. As a unifying thread across different types of systems, functional modeling enforces consistency across models, captures system behaviors to enable simulation modeling, reduces premature commitments and decisions, enables visibility across all aspects of a model, and possesses the flexibility to rapidly adapt to changes in stakeholder defined system requirements or new problems.

Current research into development of executable architectures highlights the importance of enforcing consistency within architecture models. The emphasis on Parametric Diagrams in both IBM Harmony and INCOSE OOSEM, as well as the use of CORESim in Vitech's MBSE methodology demonstrate the importance and utility of such an approach. However, while executable architectures can provide tremendous value, they are constrained by the level of detail in any associated architecture product, and therefore may not provide an adequate level of detail to fully analyze the system of interest. Ge, Hipel, Yand, and Chen (2013) highlight several of the issues associated with the current implementation of executable architectures, stating, "current executable architecture modeling efforts rely heavily on static architectural models or views of architectural descriptions." Similar limitations are noted in Wang and Dagli (2008) who use colored petri nets to realize a discrete event model based on SysML products. Numerous similar applications exist, each of which emphasizes that executable architecture approaches demonstrate tremendous value, especially by identifying capability gaps and redundant physical elements. Kim, Fried, Menegay, Soremekun, and Oster (2013) present a similar approach for the automated generation of Parametric Diagrams and even note, as is emphasized in this dissertation, that subsequent research should focus on the definition of detailed performance models that can consider system operation at multiple levels of abstraction. While this is a promising research direction,

current implementations of executable architectures are incapable of considering environmental and operations factors that may impact system performance but are not necessary elements of system design. Accordingly, this work recognizes the value of executable architecture focused research but, per the limitations associated with such architecture approaches, focuses on the development of architecture products that are capable of considering alterations to the system environment as well as system operational implementation, two major drivers of system performance that, currently, cannot be modeled in sufficient detail utilizing an executable architecture approach.

The adoption of SysML by the MBSE community is a reaction to the architecture challenges associated with development of proper functional and physical architectures. Presentation of the existing MBSE methodologies, as well as each of the SysML diagrams demonstrate that functional modeling through the use of SysML enforces consistency, captures system behavior, reduces premature commitments, and enables visibility. SysML has demonstrated promise that makes it suitable for application throughout system development, a point emphasized by Liston, Kabak, Dungan, Byrne, Young, and Heavey (2011, 300), who state, "On review of existing research in the area and the experiences gained while using the language, it is proposed that there is potential for using SysML as a common thread that could underlie all the activities undertaken in a simulation study from the initial requirements gathering phase through defining the conceptual model and on to the development of the simulation model." The authors also emphasize that while SysML is a tremendously rich language that shows promise for development of external simulation it is also inherently limited by the freedom given to the user (which introduces the possibility for misalignment with external models) as well as the substantial learning curve associated with SysML (estimating that at least 1.5 months of dedicated work is required to achieve a basic level of competency). While this is a significant learning curve, the authors note that it is not dissimilar from most other languages, and if it is utilized properly it has the potential to be used in support of discrete event simulations and "would provide a common language, which has been noted to be lacking in this domain" (Liston, Kabak, Dungan, Byrne, Young, and Heavey 2011, 303).

As an additional note, none of the research presented to this point has made it clear that the use of SysML creates the flexibility to rapidly adapt to changing problems and stakeholder requirements. That issue is currently a major focus within the MBSE community. Balestrini-Robinson, Freeman, and Browne (2015) develop a framework and interface for rapid generation of SysML products based on stakeholder interaction. While the interface is currently unable to generate visual representations of the SysML diagrams (a limitation that is currently being addressed by the authors), it defines a computer interface that rapidly creates and alters SysML products based on changes in stakeholder inputs. Furthermore, the authors emphasizes that the use of SysML is ideal for creation of architecture products due to its widespread acceptance, well-defined foundation, and its ability to represent both system performance and system interactions. Research into the use of SysML diagrams to rapidly incorporate stakeholder input is outside the scope of this research but is certainly an enabler of the methodology developed in this work. Pending further development, the use of a decision support tool to generate SysML products based on stakeholder input may be the first step in the initiation of the generic systems engineering process.

b. SysML and Simulation Linkage

Given the importance of SysML to the MBSE community and the focus within this dissertation on the specification of the appropriate usage of SysML products to link architecture and analysis it is important to review past work discussing the utilization of SysML products in the development of simulations.

Johnson (2008) presents a demonstration of the use of graph transformations to enable development of continuous dynamics models in Modelica based on SysML products. As the term continuous dynamics implies, the work focused exclusively on the physical domain, but provides a valuable demonstration of the potential to translate SysML representations into another modeling program. Cao, Liu, and Paredis (2011) extend this approach to a far more complex mechatronic system, reinforcing the potential to expand SysML products to physical modeling programs. Qamar, During, and Wikander (2009) similarly demonstrate that SysML can be linked to Simulink to

facilitate communication with stakeholders early in the design of a mechatronic system. Palachi, Cohen, and Takashi (2013) establish a similar linkage of SysML to Simulink and extend the code generation to both continuous and discrete modeling approaches. Spangelo et al. (2013) also present a similar demonstration where SysML diagrams are the basis for the development and analysis of more detailed models, in this case for a small satellite. While this work acknowledges the need to conduct operational modeling as well as physical modeling, it focuses on the utilization of Parametric Diagrams to conduct this operational modeling. The research focuses on one variable at a time changes to values in Parametric Diagrams, thereby restricting the analysis done for each potential system.

Cao, Liu, Fan, and Fan (2013) present another example of developing physical models for mechatronic systems using SysML. That work emphasizes the current direction of many relevant projects linking SysML to external simulations, specifically stating that "only the physical part of the mechatronic system is considered" and specifically scoping out control and behavior of the system. Note that this is not a negative development. In order to fully realize the benefits of SysML as a standardized architecture development language it must be linked in an executable fashion and the work referenced in this section demonstrates that such a linkage is possible from a physics based perspective. Huang, Ramamurthy, and McGinnis (2007) expand this work, demonstrating a procedure for the development of manufacturing simulation models based on SysML products. Huang (2011) expands further and develops discrete event logistics system simulations based on SysML products. That research represents one of the most substantial developments in the execution of SysML products to examine system performance. In particular, it makes a substantial contribution to the number of system states that are typically considered when systems are architected from a software perspective by utilizing internal block diagrams to fix the interactions between system components as state dependent characteristics of each component of the system. This substantially reduces the number of system interactions must be present in any subsequent discrete event model. Bataresh and McGinnis (2012) present a similar approach and create a discrete even model of a manufacturing system in Arena based on

SysML products. However, while substantial work is being done to consider external simulations after SysML products are created, past research has focused on isolated cases with a limited number of variables. It is certainly valuable to demonstrate how individual SysML Diagrams may be used to support the development of external models, but a more comprehensive framework is needed that emphasizes the need for detailed operational simulations that consider system design parameters, system components interactions, the impact of alterations to system operation, and the impact that the external environment may have on system performance. Further, no current MBSE research discusses appropriate integration of simulation model analysis results into subsequent iterations of SysML system architecture products.

c. Design and Analysis of Large Scale Simulation Experiments

Recent work at the Simulation Experiment & Efficient Designs (SEED) Center at the Naval Postgraduate School focuses on the proper design and analysis of large-scale simulation experiments (the term large scale, as generally referenced in Lucas at al. (2015), classifies simulations examining hundreds of input variables). Sanchez et al. (2012) detail that large scale simulation work and present fundamentals for the selection of an appropriate experimental design for a large scale simulation experiments (generally Latin hypercubes are shown to be good all-purpose designs), techniques for the utilization of fractional factorial designs to supplement traditional implemented designs (such as central composite designs), and sequential screening approaches to designs that may be implemented when the number of factors is very large. The utility of this approach to large scale simulation experiments is demonstrated in an analysis of U.S. Army Unmanned Aerial Vehicles, where descriptive statistical analysis, interactive regression analysis, regression trees, and contour profilers are shown to be useful analysis techniques for the analysis of unrealized systems using a large scale simulation experiment. The results of the analysis directly changed procurement decisions made by the U.S. Army. The principles presented in that work have been applied successfully in multiple domains to conduct analysis of complicated systems characterized by a very large number of components. Kaymal (2013) investigates the operational effectiveness of a surface combatant in an anti-surface warfare environment, Parker (2015) investigates

the development of future Marine Corps amphibious capabilities, Treml (2013) investigates the development of the U.S. Army Future Ground Combat Vehicle, and Wakeman (2012) analyzes key leader engagements using discrete event simulation. This is by no means a comprehensive list; rather it is an example of the utilization of large scale simulation experiments to support analysis of: a Navy system (Kaymal), a Marine Corps system (Parker), an Army system (Treml), and a social system (Wakeman). It is possible to develop and analyze high quality simulation models for a wide variety of systems without the use of MBSE (or systems engineering in general). Accordingly, it is vitally important to emphasize the role of MBSE from a simulation perspective and to identify the similarities and differences between MBSE approaches and fundamentals and currently established simulation development techniques.

The examples above make use of system, operational, and environmental variables; in many cases explicitly developing systems that are robust to uncertainties in the environment. Consideration of that broad range of variables and the use of design experiments facilitates trade space analysis. Links to these theses, methodological and application papers, as well as software and spreadsheets for constructing large-scale design can be found at the SEED Center's web page harvest.nps.edu.

d. MBSE Focused System Analysis and Trade Space Exploration

Development of models and simulations during the conceptual design phase is often challenging due to the immense number of potential system configurations. This issue is addressed in detail in Chapter III, but several guidelines have been established in recent MBSE research. In particular, Haveman and Bonnema (2015) survey modeling and simulation in early stage systems engineering and conclude that discrete event simulations are particularly well suited to conceptual systems. This could be extended to include low fidelity agent based models, as the authors advocate the use of discrete event models by noting that, "we are often more interested in the system as a whole than exploring physics based principles." Humman and Madni (2014) support using agent based models early in the system design cycle, presenting two case studies that detail successful use of agent based models to support early stage systems engineering

decisions. Sha, Le, and Panchal (2011) develop a basic agent based model based on SysML products that represents products as directed graphs. Acheson, Dagli, and Kilicay-Ergin (2013) also present a demonstration of the utilization of model-based architectures to ensure proper definition of agent based models for systems of systems. Wang and Dagli (2011) present a similar demonstration for the use of discrete event simulation to model a network sensor system. MacCalman (2013) presents the simultaneous analysis of agent based simulations developed by McKeown (2012) and Yoosiri (2012) as well as a spreadsheet based model developed by Ashpari (2012). MacCalman, Beery, and Paulo (working paper) use the same simulations to formally define a tradespace visualization approach. This research does not intend to expand the body of knowledge associated with these simulation models, rather the discussion of the alternative modeling approaches is included to demonstrate to the unfamiliar reader the breadth of potential modeling approaches that have been applied successfully in support of MBSE. Readers with limited experience developing and implementing models and simulations should refer to Law (2014) for an overview of simulation basics, simulation software alternatives, basic probability and statistics, model construction guidelines, output analysis, and a detailed review of both discrete event and agent based models.

Of particular interest within Law (2014) is the creation and management of an assumptions document. Also reviewed and summarized more briefly in Law (2009), the assumptions document is presents "all concepts, assumptions, algorithms, and data summaries" that reduce potential communication issues. The assumption document is provides a "blueprint" that "represents the model developers' initial thoughts on the form the model will eventually take" (Law 2009, 29). An assumptions document includes a list of system processes, subsystems, simplifying assumptions, limitations, input data, and information sources to aid in communication with stakeholders. In this way the assumptions document shares many of the same goals of SysML product development. Several notable differences demonstrate the value of SysML product development. First, capture of system information using SysML compatible software ensures consistency and traceability between multiple models. For instance, if a function is developed and allocated to a subsystem in a SysML Activity Diagram but that subsystem is not

associated with a system that also performs that function, the inconsistency will be immediately visible in a SysML Internal Block Diagram. Perhaps more importantly, while processes and layouts trace to system requirements or performance measures within an assumptions document, that traceability cannot be mandated or enforced. Utilization of SysML compatible software ensures that system processes and layouts are directly linked to system requirements and performance measures and rapidly and consistently updates system requirements based on changes to system structure. This does not suggest that the creation of an assumptions document is inappropriate or invaluable, rather it emphasizes that the use of detailed model-based systems engineering architecture products allows for more detailed, relationships to be modeled in an architecture program that ensures traceability and consistency as well as rapid updating and reuse.

Law (2009) also draws a loose analogy between assumptions documents and conceptual models. The most notable developments regarding conceptual models, to include the verification and validation of conceptual models, is summarized in Sargent (2013), where a conceptual model is defined as "the mathematical/logical/verbal representation (mimic) or the problem entity developed for a particular study" (Sargent 2013, 323). The purpose of the conceptual model is to establish a linkage between the real system and a more detailed computerized model that can be validated by "determining that the theories and assumptions underlying the conceptual model are correct" (Sargent 2013, 324). This is analogous to the approach that will be advocated by this dissertation, which suggests that SysML architecture products can be used as a linkage between real systems and more detailed simulation models. However, as with the assumptions document, the conceptual model is often a static narrative model that describes a system, rather than a dynamic architecture formulation that establishes interactive, rapidly configurable relationships between system functions and components. While the general process advocated by Sargent (2013) that uses an intermediate (or conceptual model) to link the real world to the simulation world is a hugely powerful and appropriate paradigm, the power and richness of SysML compatible software allows for

the utility of this linkage to be increased and the codification of this linkage is a major focus of this research.

While both Law (2009) and Sargent (2013) emphasize the importance of describing a system to be modeled in a formalized way prior to the development of more detailed computer models, it should also be noted that the system must also be described in the context of its intended operation and environment. As mentioned previously, this has been one of the major limitations associated with current executable architecting approaches. The importance of using models and simulations to consider the entire system, as well as its external environment, is the basis for robust design research (see Sanchez (2000) and Montgomery (2012)). That importance is summarized by Giammarco and Auguston (2013), who define two key principles for system modeling, specifically.

1. In addition to modeling the behavior of the system along with its interfaces to external systems, also model the behavior of the environment in which the system operates
2. Model component interactions abstractly and separately, rather than instantiated in specific use cases (Giammarco and Auguston 2013, 280)

While the authors established those principles to support development of a specific MBSE architecture framework, when considered with the work of Haveman and Bonnema (2015) it is evident that development of external models and simulations (both discrete event and agent based) that consider the interactions between system components as well as interactions between the system and its environment is necessary to examine system performance during the conceptual design phase. Accordingly, discrete event and agent based models are recommended for use in conjunction with this research and users should ensure that the two key principles presented above are used as guidelines during simulation development. Furthermore, the work of MacCalman et al. (2015) presents a demonstration of the potential value of building external models and simulations based on stakeholder analysis and system architecture development. That research closely aligns with the approach advocated by this dissertation and provides an in depth demonstration of the use of new experimental design techniques to design and exercise an agent based simulation of a U.S. Army infantry squad. That research further explores the development of tradespace visualization tools and provides a comprehensive overview of the value of such tools as well as guidelines for development of tradespace

visualization software. That work provides a complete description of the utility of complete tradespace exploration and makes a substantial contribution to the system analysis domain, as outlined earlier in this dissertation. However, none of the recent advances in system analysis and tradespace visualization provide a roadmap for the utilization of system architecture products to develop external simulation models.

This section familiarizes the reader with current MBSE focused architecture research as well as current simulation development research. In general, this research demonstrates the appropriate definition of model-based system architecture products given that they will subsequently be used to develop detailed models and simulations. Recent MBSE research has progressed to the point that development of such architecture products is possible; however recent research focuses almost exclusively on the formalization of those descriptive architecture products. Furthermore, simulation and system analysis research is conducted successfully and products such as assumptions documents and conceptual models are used in lieu of detailed system architectures. However, the recent advances within the MBSE community to formalize SysML products now makes it possible to utilize those products (in much the same manner as assumptions documents and conceptual models) to better define necessary system functions and components and to more rapidly integrate system analysis results into formal system descriptions. That definition and integration is the major focus of the MBSE MEASA. A review of the most widely known architecting approach for DOD systems is necessary before demonstrating the utility of the MBSE MEASA.

5. Department of Defense Architecture Framework

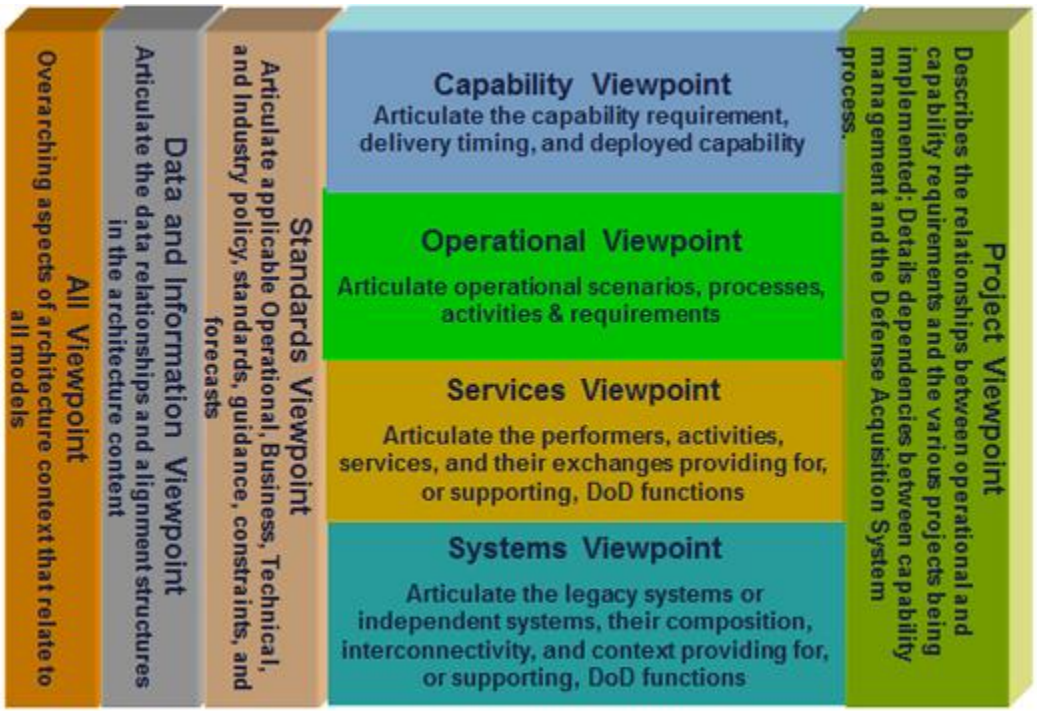
The Department of Defense Architecture Framework (DoDAF) is DOD's framework to enable the development of system architectures and share information across organizational boundaries. The current DoDAF release, version 2.02 (released in August 2010) emphasizes the development of architectural "data" rather than the production of architectural "products," although the production of architectural views is still the primary output of implementation of DoDAF. These architectural views are capability views, data and information views, operational views, project views, services views, standards views, or systems views, the production of which provides in depth

information regarding specific areas of interest while maintaining a comprehensive description of the full system enterprise. As presented in Department of Defense Chief Information Officer (2015), "each viewpoint has a particular purpose, and usually presents one or combinations of the following:

1. Broad summary information about the whole enterprise (e.g., high-level operational concepts
2. Narrowly focused information for a specialist purpose (e.g., system interface definitions)
3. Information about how aspects of the enterprise are connected (e.g., how business or operational activities are supported by a systems, or how program management brings together the different aspects of network enabled capability)"

The broad range of DoDAF views enables a multitude of potential mechanisms for information capture and communication (either enterprise wide or specific). Figure 18 presents a brief summary of the data captured in each DoDAF viewpoint.

Figure 18 DoDAF Viewpoints



Source: Department of Defense Chief Information Officer. 2015. "DoDAF: DOD Architecture Framework Version 2.02 DOD Deputy Chief Information Officer." August 11. http://dodcio.defense.gov/Library/DoDArchitectureFramework.aspx

Utilization of DoDAF views in system development results in a definition of a coherent system model that can be viewed from multiple perspectives. Piaszczyk (2011) provides a comprehensive overview of employing a model-based systems engineering approach utilizing DoDAF products. Review of that work demonstrates that development of DoDAF views is similar in intent to development of SysML Diagrams. There are several major differences between DoDAF and SysML that are relevant to this research. First, utilization of DoDAF (from a practical perspective) is obviously restricted to application to DOD systems. Second, DoDAF is tailored for application at the program level to facilitate communication between engineers, program managers, stakeholders, and outside businesses, which necessarily means that it has levels of complexity that may be beyond the scope of this research. Garrett, Anderson, Baron, and Moreland (2011) summarize the true utility of DoDAF views, stating that development of DoDAF viewpoints "provides a means for the program manager and systems engineer to work with the stakeholder in translating the architecture views into verifiable requirements." This dissertation research intends to facilitate communication between system architects and system analysts, and while it may be useful as an expansion of some portions of DoDAF, it is not intended to be as broadly applicable as DoDAF. Finally, the DoDAF Systems Viewpoints, which describe systems and interconnections between systems, adhere to a similar perspective as the industrial MBSE methodologies presented earlier. Specifically, creation of DoDAF Systems Viewpoints, even when integrated with other DoDAF Viewpoints, still focus development on functional architecture, physical architecture, and an executable architecture that checks for consistency between those architectures. DoDAF is not specifically configured to support development of external simulation models. As Garrett, Anderson, Baron, and Moreland (2011) state, "The development of the system architecture and corresponding executable models provide a way to capture the definition of the system requirements and functional and physical architectures that define the functions, allocated, and product baselines." This emphasis on consistency within existing architecture models is certainly extremely valuable, however it does not allow for a complete examination of system performance. The MBSE MEASA presented in this research defines a roadmap for utilizing standardized, accepted

system architecture products to develop external models and simulations that can be used to conduct detailed analysis of system performance, which in turn can be used to develop more complete systems requirements.

Substantial research in academia, industry, and within DOD defines a model-based approach to system development and design. The inclusion of both architecture development and system performance modeling in each of the widely used MBSE methodologies demonstrates that these processes are essential to development of new systems using MBSE focused development. The substantial effort dedicated to evaluation of the utility of SysML products in recent systems engineering conference proceedings and journal articles demonstrates their importance to the systems engineering community as well as their acceptance as the standard starting point for MBSE focused development of a new system. The system analysis community has developed system description approaches (most notably in the form of assumptions documents and conceptual models) that describe the relationship between the real world and detail simulation models. However, while there has been industrial research that developed MBSE methodologies and academic research that defined methods for generation of simulations based on SysML products, neither the industrial or academic community has defined an end-to-end integrative methodology that establishes linkage between model-based architectures and detailed system operational, physical, and cost models. The MBSE MEASA expands the state of the art in MBSE by defining a comprehensive framework that uses SysML based system architecture products as the basis for external simulation models and integrates the results of the analysis of those models into future iterations of the system architecture for a wide range of potential analysis results. This expands the utility of the current systems development approach advocated by IBM, INCOSE, Vitech, and DoDAF, where the execution and evaluation of an allocated architecture is often the endpoint of the system development process. The MBSE MEASA prescribes the use of architecture products that characterize operational, physical, and cost models. By leveraging existing state-of-the-art methods in the design and analysis of large-scale simulation experiments, this expands the reach of any current MBSE methodology by considering not only the ability of a physical system configuration to satisfy a given set of functions but also

considering the interactions between the system and the environment, variations to system operations, and interactions between system components. Further, the consideration of each of these variable types, as well as the interactions between those variables, allows the MBSE MEASA to uniquely define a procedure for the integration of analysis results into future iterations of the system architecture.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MODEL-BASED SYSTEMS ENGINEERING METHODOLOGY FOR EMPLOYING ARCHITECTURE IN SYSTEM ANAYLSIS DEFINITION

This research develops an MBSE MEASA that establishes a linkage between the system architecture and system analysis domains by defining the proper use of external models and simulations, based on SysML architecture products, to develop more complete system requirements. Given that processes such as IBM Harmony for Systems Engineering, INCOSE Object Oriented Systems Engineering, Vitech's Model-Based Systems Engineering Methodology, NASA's Jet Propulsion Lab State Analysis, Dori's Object-Process Methodology, and Weilkiens' Systems Modeling Process have established frameworks for executing the systems engineering process through a model-based approach, the MBSE MEASA can be considered an extension of those processes that facilitates detailed analysis of system performance earlier in the systems engineering process. Specifically, the MBSE MEASA is intended to enable development of complicated, large scale systems effectively through analysis of models and simulations that consider not only system design attributes (as is done in each of the MBSE methodologies presented in the previous chapter) but also environmental and operational factors during system conceptual design. Development of the MBSE MEASA must fit within the context of a general MBSE process model and augment the capabilities already provided by existing process models. More specifically, an MBSE MEASA must be developed within the context of the previously stated systems engineering process characteristics and should be shaped to satisfy the following goals, developed by synthesizing the previously presented benefits of both systems engineering and model-based systems engineering:

1. The process will result in learning, continuous improvement, discovery of requirements, discovery of system properties, and discovery of system behavior
2. As a result, the process will reduce uncertainty about a system and serve as a framework and mechanism that drives system development towards a solution that best satisfies predefined system requirements

The MBSE MEASA satisfies each of those goals, given implementation within a quality systems engineering process (integration of the MBSE MEASA with the model-based systems engineering processes/methods described previously may be particularly useful, depending on the system of interest).

A. SYSTEMS ENGINEERING PROCESS DEFINITION

As demonstrated in Chapter II, numerous systems engineering process models exist, and the MBSE MEASA is usable within the context of any of those models. Rather than choose a specific systems engineering process model and implement the MBSE MEASA within that model, the general systems engineering process, comprised of the following steps, is considered and used as the basis for development of the MBSE MEASA. Recall that all systems engineering processes should be iterative, in particular system analysis results should be used to inform the system stakeholder, which should then promulgate down to subsequent requirements and architectures.

1. Problem Definition

i) Stakeholder Analysis
ii) Requirements Identification

2. System Design

i) Functional Architecture Development
ii) Physical Architecture Development
iii) Allocated Architecture Development
iv) Modeling and Simulation

3. System Analysis

i) Assessment of System Designs
ii) Cost Analysis

4. System Implementation

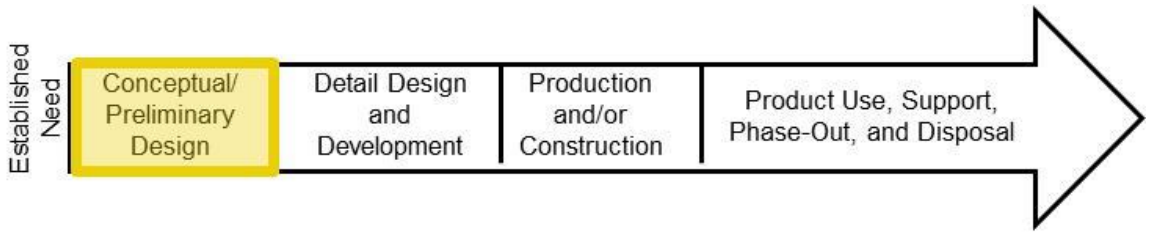
i) System Production
ii) System Deployment
iii) System Operation
iv) System Disposal

The generic systems engineering process generalizes the steps outlined in each of the systems engineering processes presented earlier. This generalization is advantageous for two reasons. First, it establishes a general process that the MBSE MEASA supports. Second, when each step of the process is analyzed, a defined set of products are established that demonstrate the value added by completing each step of the process.

Two assumptions are important before proceeding. Given that this research is focused on development of an analysis methodology, particular attention is given to the first three major sections of the general systems engineering process: Problem Definition, System Design, and System Analysis. This is due to the focus of the MBSE MEASA, specifically the intent for the methodology to be used for definition, design, and analysis of large scale, complex systems. While system implementation is extraordinarily important, it is distinct from system development and is therefore more appropriate for discussion in project management literature than it is for inclusion in development of a systems engineering process model or systems engineering analysis methodology. Note that this does not reduce the importance of iteration of the process; rather it means that iteration occurs within the steps of the process, as well as at the end of system analysis (rather than system implementation), to inform subsequent iterations of stakeholder analysis or requirements identification.

Furthermore, the MBSE MEASA assumes that an initial stakeholder analysis (the first step of Problem Definition) is complete. The importance of quality stakeholder analysis should not be understated. As Trainor and Parnell (2011, 297) state, "a great solution to the wrong problem is…wrong." Improper problem definition results in substantially diminished impact for system design, system analysis, and system implementation. Balestrini-Robinson, Freeman, and Browne (2015) outline current MBSE related research in this area. Because the MBSE MEASA focuses on the conceptual design phase, stakeholder analysis is not the focus of this dissertation. Figure 19 provides a high level overview of a generic system life cycle and highlights the portion of the system life cycle of interest to this research (note that Conceptual/Preliminary Design is only initiated subsequent to development of an Established Need):

Figure 19 Generic System Life Cycle



Adapted from: Blanchard, Benjamin S., and Wolter J. Fabrycky. 2010. *Systems Engineering and Analysis*, 5th ed. Upper Saddle River, NJ: Pearson Prentice Hall

It is now possible to state the set of products created during Problem Definition, System Design, and System Analysis. Synthesizing the set of products recommended for development in Blanchard and Fabrycky (2010) and Buede (2009), Problem Definition results in: a Defined Problem, a Defined System Boundary, a Defined System Objective, and Defined System Requirements. System Design results in Defined Functional Behaviors, Defined Functional Performance, Defined Allocation of Requirements to Functions, Defined Candidate Physical Solutions, and a Defined Model of Physical Solutions. System Analysis results in: Evaluation of Candidate Physical Solutions and an Assessment of Physical Solutions' Satisfaction of System Requirements. If all of these products are generated (this is most easily accomplished by adherence to the general systems engineering process), it is likely that any system development decisions will be made in support of stakeholder identified needs/requirements. The MBSE MEASA supports each of the above products, which facilitates use of the MBSE MEASA in conjunction with any systems engineering process model (since all process models follow the same generic systems engineering process and therefore all process models will create the same set of products outlined above).

The MBSE MEASA enables realization of the intended benefits of MBSE. Creation of each of the products outlined above ensures that the MBSE MEASA supports a generic systems engineering process, while additional criteria assess the ability of the MBSE MEASA to realize the intended benefits of MBSE. The four intended benefits of MBSE developed by Friedenthal, Griego, and Sampson (2007) (these intended benefits are shown as bullets 1, 2, 3, and 4), along with related criteria developed by the author

(shown as bullets “a” and “b” for each intended benefit) that can be used to assess the utility of the MBSE MEASA:

1. Improved communications among the development stakeholders

(a) Does the MBSE MEASA explicitly incorporate stakeholder input?

2. Increased ability to manage system complexity by enabling a system model to be viewed from multiple perspectives, and to analyze the impact of changes

(a) Does the MBSE MEASA allow the system model to be viewed from multiple perspectives?
(b) Does the MBSE MEASA incorporate a method for analyzing the impact of changes to the system design?

3. Improved product quality by providing an unambiguous and precise model of the system that can be evaluated for consistency, correctness, and completeness

(a) Does the MBSE MEASA provide an unambiguous and precise model of the system?
(b) Can the models developed in the context of the MBSE MEASA be evaluated for consistency, correctness, and completeness?

4. Enhanced knowledge capture and reuse of information by capturing information in more standardized ways and leveraging built in abstraction mechanism inherent in model driven approaches. This is turn can result in reduced cycle time and lower maintenance costs to modify the design

(a) Does the MBSE MEASA capture information in standard ways?
(b) Does the MBSE MEASA enable reduced cycle time and lower maintenance costs to modify system designs?

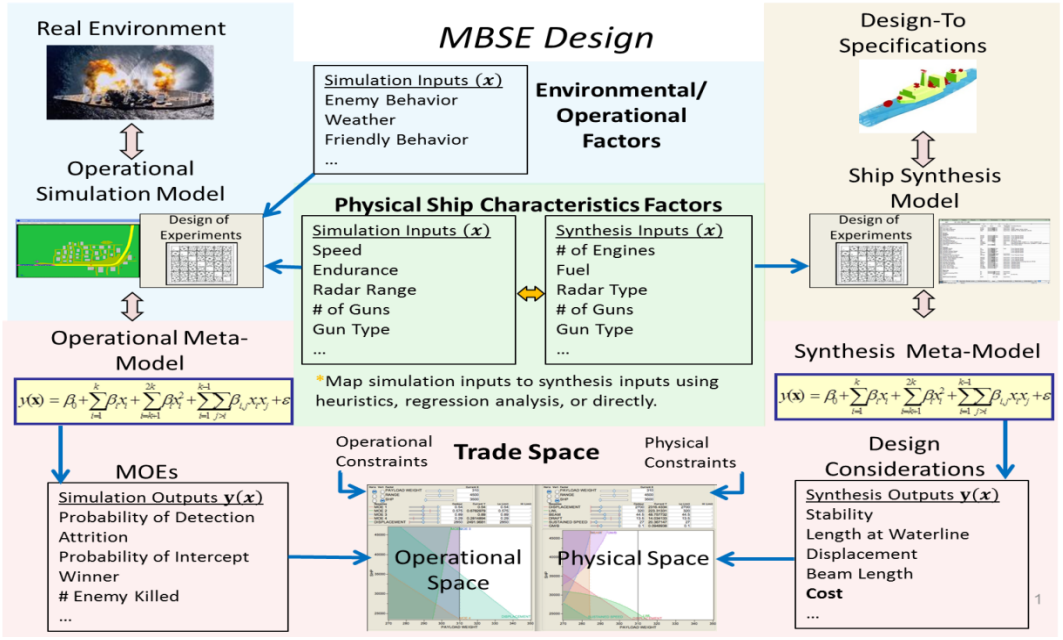
Note that the fifth intended benefit of MBSE, the “improved ability to teach and learn systems engineering” is not included because it relates to the larger intended benefits of MBSE and not to the benefits of MBSE in terms of system definition, design, and analysis. Given these stated criteria, SysML (which was developed to support many of these goals) should be incorporated with the MBSE MEASA.

B. MBSE MEASA PRESENTATION

1. Analysis Methodology

Figure 20, from MacCalman (2013) and expanded in MacCalman, Beery, and Paulo (working paper), provides a starting point for identifying the characteristics of an MBSE based analysis methodology. It is a desirable starting point for this research because it establishes the formal linkage between operational need and physical system configuration that should be the focus of any MBSE based analysis methodology. Note that Figure 20 uses the term "MBSE Design" as a description but the approach is termed "Analysis Methodology" in the context of this research.

Figure 20 Analysis Methodology



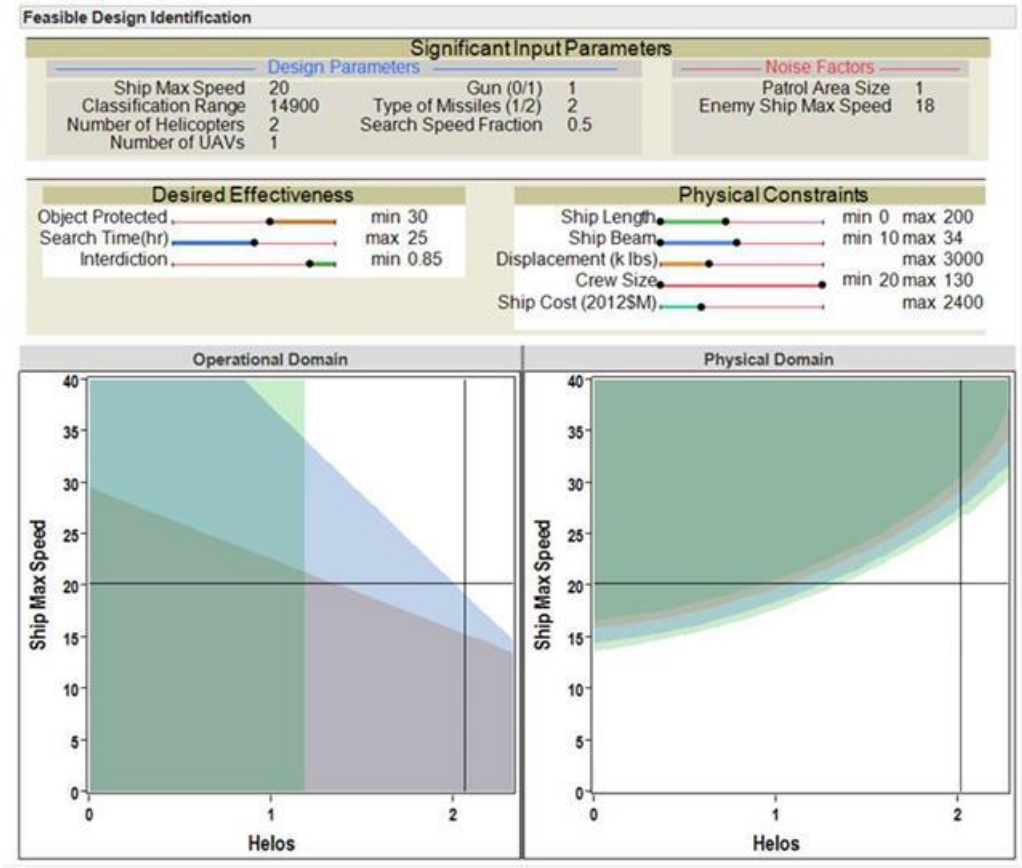
Source: MacCalman, Alexander D. 2013. "Flexible Space-Filling Designs for Complex System Simulations." Ph.D. Dissertation, Naval Postgraduate School.

The analysis methodology shown above formally defines the methodological building blocks inherent to any MBSE based analysis process. Because the process provides such a concise definition of many aspects of analysis and MBSE, it is used as the basis for the development of an MBSE MEASA. The analysis methodology

emphasizes that both operational simulation models and ship synthesis models are built using a common set (or at least a set that can be mapped) of inputs. The analysis methodology also suggests that those models be represented using regression meta-models to simultaneously visualize the Operational Space and Physical Space. That simultaneous visualization is shown in MacCalman, Beery, and Paulo (working paper) and an example (presented in Figure 21) highlights the value of such an approach. The example is based on operational simulations presented in McKeown (2012), who developed the Anti-Surface Warfare model, Yoosiri (2012), who developed the Maritime Interdiction model, Ashpari (2012), who developed the Search and Rescue model, and Lineberry (2012), who developed the cost model, and assumes that the system under consideration is a naval ship, with operational constraints imposed for various MOEs, in this case: Objected Protected, Search Time (hr), and Interdiction. There are also system constraints imposed for: Ship Length, Ship Beam, Displacement (k lbs), Crew Size, and Ship Cost (2012$M). Below each of these lists of constraints are Operational and Synthesis trade spaces. These trade spaces represent two dimensional projections of the overall potential trade space (which exists in more than two dimensions). These trade spaces are defined by the imposed constraints, where all ship combinations that cannot satisfy a given constraint are shaded out (for example, a maximum acceptable Search Time is established at 25 hours and all ship combinations shaded in Blue in the Operational trade space are incapable of satisfying that operational constraint). The resulting white region defines an operationally feasible trade space on the left and a feasible system synthesis trade space on the right. These trade spaces can be dynamically altered based on changing constraints, and potential ship combinations can be investigated for feasibility based on those constraints. For further discussion of the utility and use of such an approach, see MacCalman, Beery, and Paulo (working paper). For recommendations regarding general development of a tradespace exploration tool as well as a list of best practices regarding implementation of such a tool see Spero et al. (2014). That work is a specific expansion of the multi-attribute tradespace exploration approach first presented in Ross (2003), which developed a normative decision making approach for exploration of multi-dimensional tradespaces. That work presents a sequential

procedure, further expanded in an application to a satellite system by Ross, Stein, and Hastings (2014) that may be used to guide the sequencing of factor examination.

Figure 21 Trade Space of Operational and System Synthesis Simulation Models

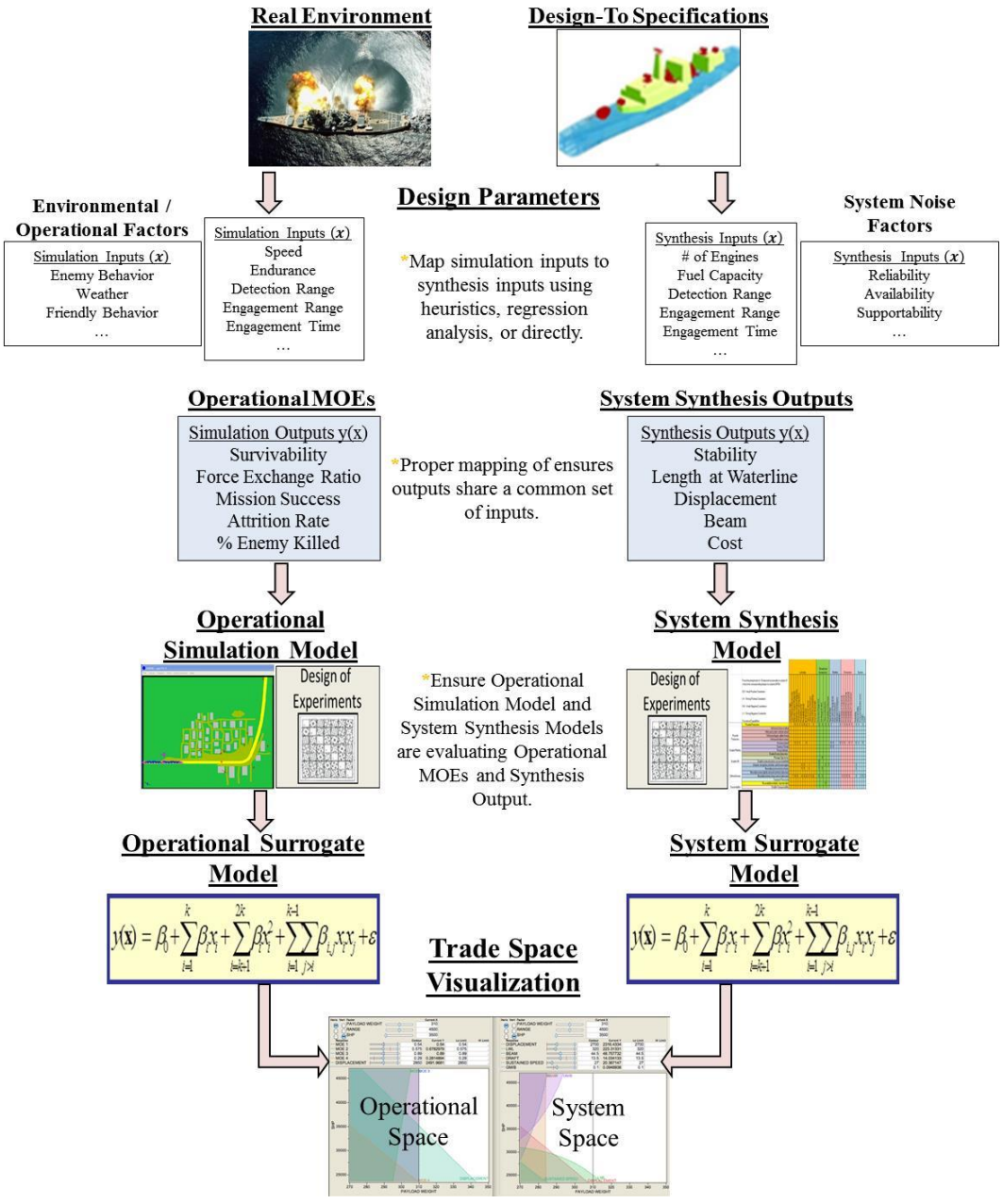


Source: MacCalman, Alexander D., Paul T. Beery, and Eugene P. Paulo. (working paper). A Systems Design Exploration Approach that Illuminates Tradespaces Using Statistical Experimental Designs.

There is utility to implementing an analysis methodology that enables the simultaneous visualization of operational and synthesis models. However, alteration of the general process presented in Figure 20 is necessary to ensure consistent, more generalizable terminology and to provide a more coherent description of the intended implementation of the approach. There is a practical segmentation of operational effectiveness models and system synthesis models because different individuals typically construct and analyze these models. Any analysis methodology that addresses both

operational effectiveness modeling and system synthesis modeling must emphasize that the models must begin with a common set of inputs (or inputs that may be mapped, as noted in Figure 21) to facilitate shared analysis of model results. Accordingly, Figure 22 presents an update version of Figure 20 that introduces these changes.

Figure 22 Revised Analysis Methodology



There are several critical differences between Figure 22 and Figure 20. The methodology presents events from a top-down perspective to communicate the intended sequencing of events. Specifically, the analysis methodology now explicitly acknowledges that the Real Environment and Design-To-Specifications are typically the start point for the development of operational effectiveness models and system synthesis models, respectively. Similarly, the Trade Space Visualization is now the clear, common endpoint of the analysis methodology, emphasizing that development and analysis of the operational effectiveness models and system synthesis models supports shared tradespace visualization. Note that, per the definitions of systems engineering process models presented earlier, iteration of the process may be necessary. In this case, the results of Trade Space Visualization should develop new system descriptions in terms of the Real Environment and the Design-To-Specifications. Figure 23, Figure 24, and Figure 25 segment the methodology to facilitate a more complete description; however before presenting that detail it is important to highlight several terminology changes from Figure 20 to Figure 22.

Figure 22 implements numerous terminology changes. The term “MBSE Design” is now “Analysis Methodology.” The altered terminology more accurately represents the intended utility because the analysis methodology is intended to be used in conjunction with previously developed SysML products (which, when combined, comprise the MBSE MEASA). The term “Physical Ship Characteristics Factors” is now “Design Parameters.” This emphasizes the generalizability of the analysis methodology and also avoids confusion with the terminology used to define synthesis models. Note that “Design Parameters” describes both Simulation Inputs, shown on the left of Figure 22, as well as Synthesis Inputs, shown on the right of Figure 22. Several Simulation Inputs and Synthesis Inputs are also updated to preserve solution neutrality. The term “Ship Synthesis Model” is now “System Synthesis Model” and the term “Synthesis Meta-Model” is now “System Surrogate Model” to emphasize generalizability. The term “Design Considerations” is now “System Synthesis Outputs” to provide a clearer linkage of the System Synthesis Outputs to both the System Synthesis Model and the System Synthesis Surrogate Model. The term “MOEs” is now “Operational MOEs” to provide a

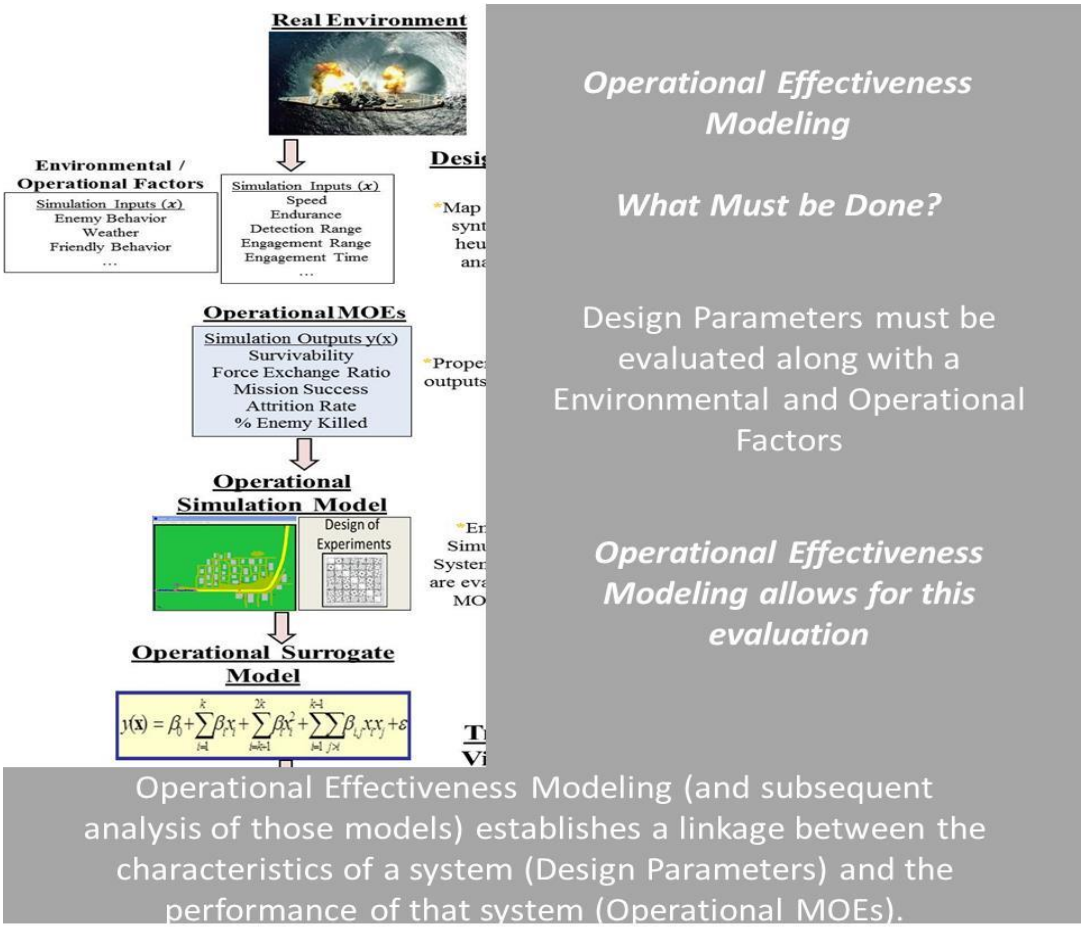
similar linkage to the Operational Simulation Model and the Operational Surrogate Model.

Examination of Figure 22 in more detail is necessary. Segmenting the figure into three distinct subsections and examining them sequentially makes this examination easier. Note that Figure 22 defines the analysis methodology in the context of a naval ship, but the process is generalizable to any large scale, complex system. This research utilizes the naval ship example to more easily demonstrate the potential utility of the analysis methodology.

Implementation of the analysis methodology begins with the development of operational simulation models (Figure 23). Defining the intended model inputs and model outputs initiates development of operational simulation. In the case an operational simulation model for a new naval vessel, the modeling inputs are segmented into two distinct categories, controllable ship design characteristics (listed in Figure 23 under Design Parameters – Simulation Inputs) and uncontrollable environmental or operational factors (listed in Figure 23 as Environmental/Operational Factors). The controllable ship design characteristics (ex: Speed, Endurance, Detection Range, Engagement Range, etc.) are evaluated across a broad range of uncontrollable environmental and operational factors (ex: Enemy Behavior, Weather, Friendly Behavior, etc.) in the operational simulation. The purpose of the operational simulation model is to establish a linkage between these model inputs to an operationally relevant set of model outputs (listed in Figure 23 as Operational MOEs). Through the use of proper experimental designs, the linkage of the model inputs to model outputs, or measures of effectiveness (MOEs), can be represented in a statistically valid surrogate model, which can subsequently serve as a surrogate to the simulation itself. Use of such a surrogate model allows for a rapid examination of the relationships between model inputs and outputs. As an example, it would be possible for a minimum acceptable performance standard to be set for one of the MOEs (ex: Attrition Rate) and the set of ships capable of satisfying that performance standard could be defined (ex: the ships with sufficient Speed, Endurance, Detection Range, Engagement Range, and Engagement Time to satisfy the standard for Attrition Rate).

Starting the design process with operational simulations is the foundation of the analysis methodology. Rather than defining a desired ship (or system) in terms of a preferred ship length, ship beam, ship displacement, radar range, number of guns, etc., and subsequently assessing the ability of that ship to meet various performance criteria, the analysis methodology advocates beginning the design process by considering the performance criteria. If done properly, the analysis methodology should prevent development of any system that does not directly support specific Operational MOEs (as well as any system that does not provide satisfactory performance with respect to each of those Operational MOEs).

Figure 23 Analysis Methodology: Operational Effectiveness Modeling



While the value of initiating the process with the development of operational simulation models may be clear, it should not be underemphasized. This ordering aligns with each of the systems engineering process models presented earlier. In particular, system development and analysis should focus on the functions that a system must perform (informally described as "what" a system must do) before exploring the set of system configurations that can perform those functions (informally described as "how" the system will be configured). While this aligns with the systems engineering processes models, the ship building community does not always practice this sequencing of operational models and system synthesis models. This issue was first identified by Frits, Weston, Pouchet, Kusmik, Krol, and Mavris (2002) and formally stated by Hootman and Whitcomb (2005, 44), who state, "the use of effectiveness analysis existed, but it was virtually decoupled from the design process." This decoupling runs the risk of entering into a sequence where physical systems are developed and subsequently analyzed to determine performance, which may result in development of systems without emphasis on functionality. Developing and analyzing operational simulation models prior to system synthesis models can mitigate this risk.

After developing and analyzing operational simulation models, the analysis methodology moves to development and analysis of system synthesis models (Figure 24). The modeling approach is nearly equivalent to the operational simulation models. Model inputs, shown under Design Parameters – Synthesis Inputs (in this case, Number of Engines, Fuel Capacity, Detection Range, Engagement Range, etc.) are linked to model outputs (shown as Synthesis Outputs – Ship Stability, Length at Waterline, Displacement, etc.). Analysis of the output results in development of a surrogate model that rapidly reproduces the results of any system synthesis model. Introduction of design standards (such as maximum acceptable ship length, maximum acceptable displacement, etc.) prompts assessment (using the surrogate models) of the feasibility of those design standards for a given set of ship characteristics (Speed, Endurance, Detection Range, Engagement Range, etc.). Linkage of modeling results is possible because the system synthesis models and the previously developed operational synthesis models have the same inputs (with some potential mapping, such as Endurance to Fuel Capacity).

Figure 24 Analysis Methodology: System Synthesis Modeling

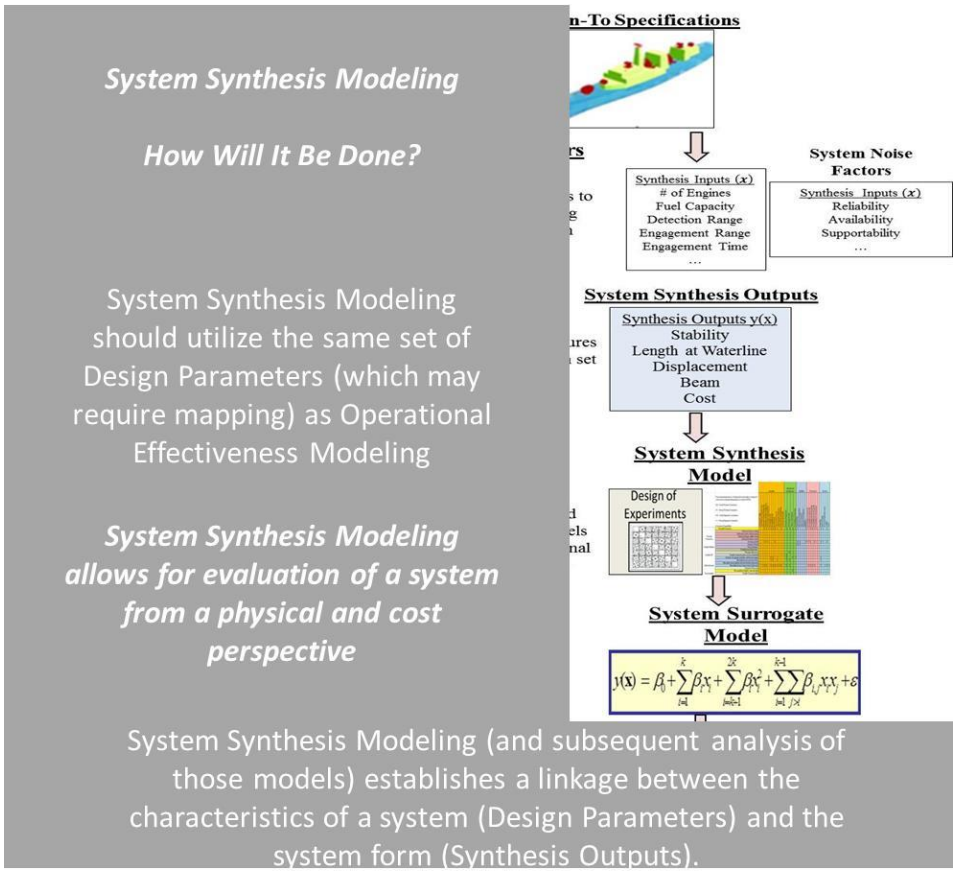
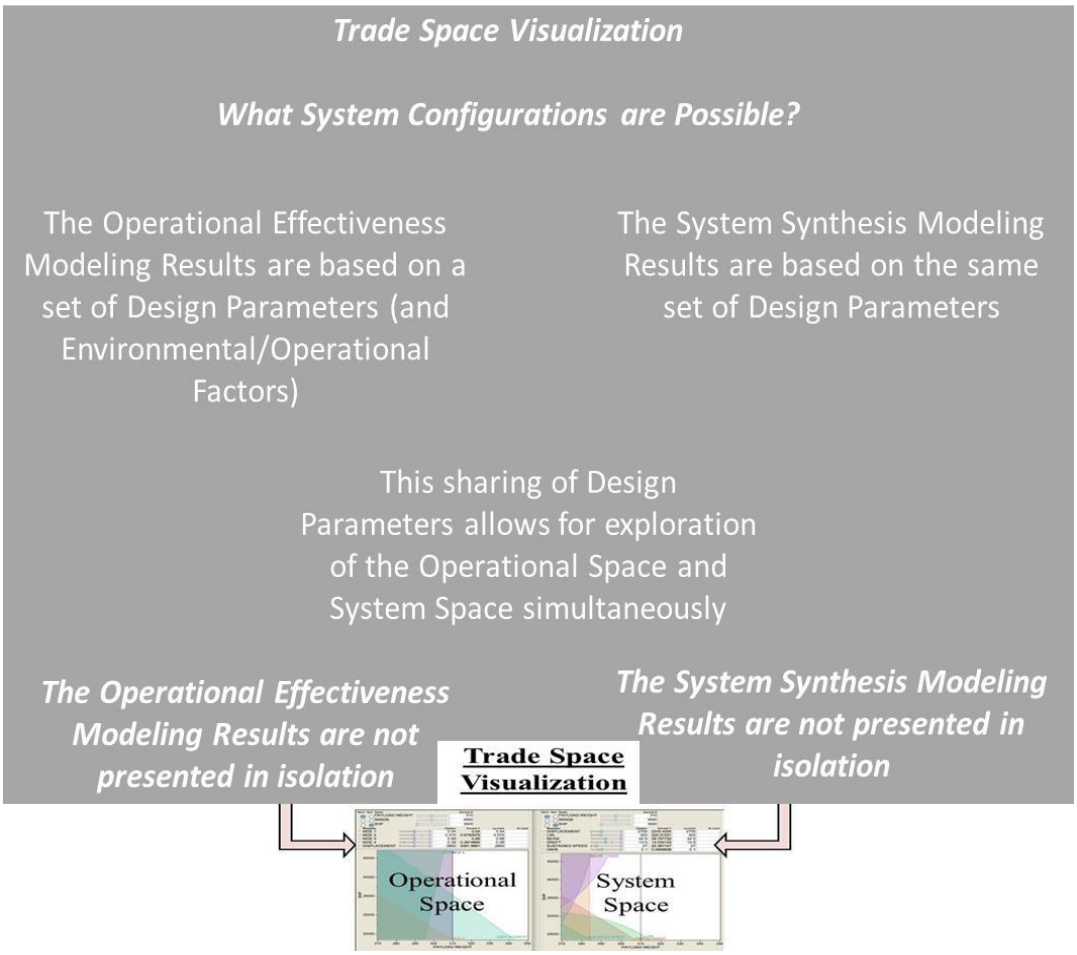


Figure 25 highlights the final step in the process, simultaneous presentation of the results of operational simulation models and system synthesis models. Simultaneous and dynamic examination of the operational and system space is possible after analysis of the modeling results and development of surrogate models. This facilitates examination of the complete trade space rather than a single design recommendation based on some form of multi-objective optimization. A set of operational constraints (performance standards) can be imposed and the set of ship combinations (in terms of Design Parameters such as Speed, Endurance, etc.) that satisfy those constraints can be defined as an operationally feasible trade space. Similarly, a set of system constraints (design standards) can be imposed and the set of ship combinations (in terms of Design Parameters such as Speed, Endurance, etc.) that satisfy those constraints can be defined as a feasible system trade space. The set of ship combinations that satisfy both the operational and system constraints can immediately be visualized and a set of feasible ship combinations can be

defined. There are three potential situations where no feasible configurations exist. First, there may be no feasible configurations in the Operational Space. Second, there may be no feasible configurations in the System Space. Finally, there may be no overlap between the feasible configurations identified in the Operational Space and the feasible configurations identified in the System Space. Two potential solutions exist in these situations. The first solution, which is far more difficult, is re-running each model for different ranges of each Design Parameter (for example, if the Speed was examined from 0 to 40 initially, it may be examined from 0 to 50 instead). This increases the size of the trade space and may increase the number of potentially feasible configurations. The second potential solution is that the operational and system constraints may be relaxed to increase the number of feasible configurations.

Figure 25 Analysis Methodology: Trade Space Visualization



While analysis of a large scale, complex system using the process defined above certainly may define a system trade space and ensure that system development decisions are not made without consideration of operational performance, the process must be expanded, clarified, and defined. Fitting the above analysis methodology process into the standard SE process assumes that several earlier tasks have already been completed, namely, that a comprehensive user requirements analysis has taken place, that a functional architecture has been developed, that a set of candidate physical architectures has been defined, and that an operational/allocated architecture that supports modeling decisions has been completed. Clear definition of how each of these systems engineering tasks integrates with this analysis methodology is a major effort of this dissertation. This research defines how each of these traditional systems engineering tasks supports and integrates with the analysis methodology presented above and, in particular, defines how various SysML products can be used to support system analysis and development. As mentioned, the use of SysML products is the major focus of the majority of the leading MBSE methodologies. This research considers development of those SysML products the primary enabler of the MBSE MEASA from a system architecture perspective. This research uses those products as a basis, segments the products according to their implementation within the generic systems engineering process, and identifies the characteristics of each system architecture product that supports the development of external models and simulations.

2. MBSE MEASA Definition

This research defines a linkage between system architecture products and system analysis products. Specifically, this research identifies a mechanism for the integration of SysML products, grouped into functional and physical architecture focused diagrams, with external models and simulations. The full description of the integration between those products comprises the MBSE MEASA. To support that development, Figure 26 presents the baseline analysis methodology again establishes a starting point for linkage of simulation models.

Figure 26 Analysis Methodology

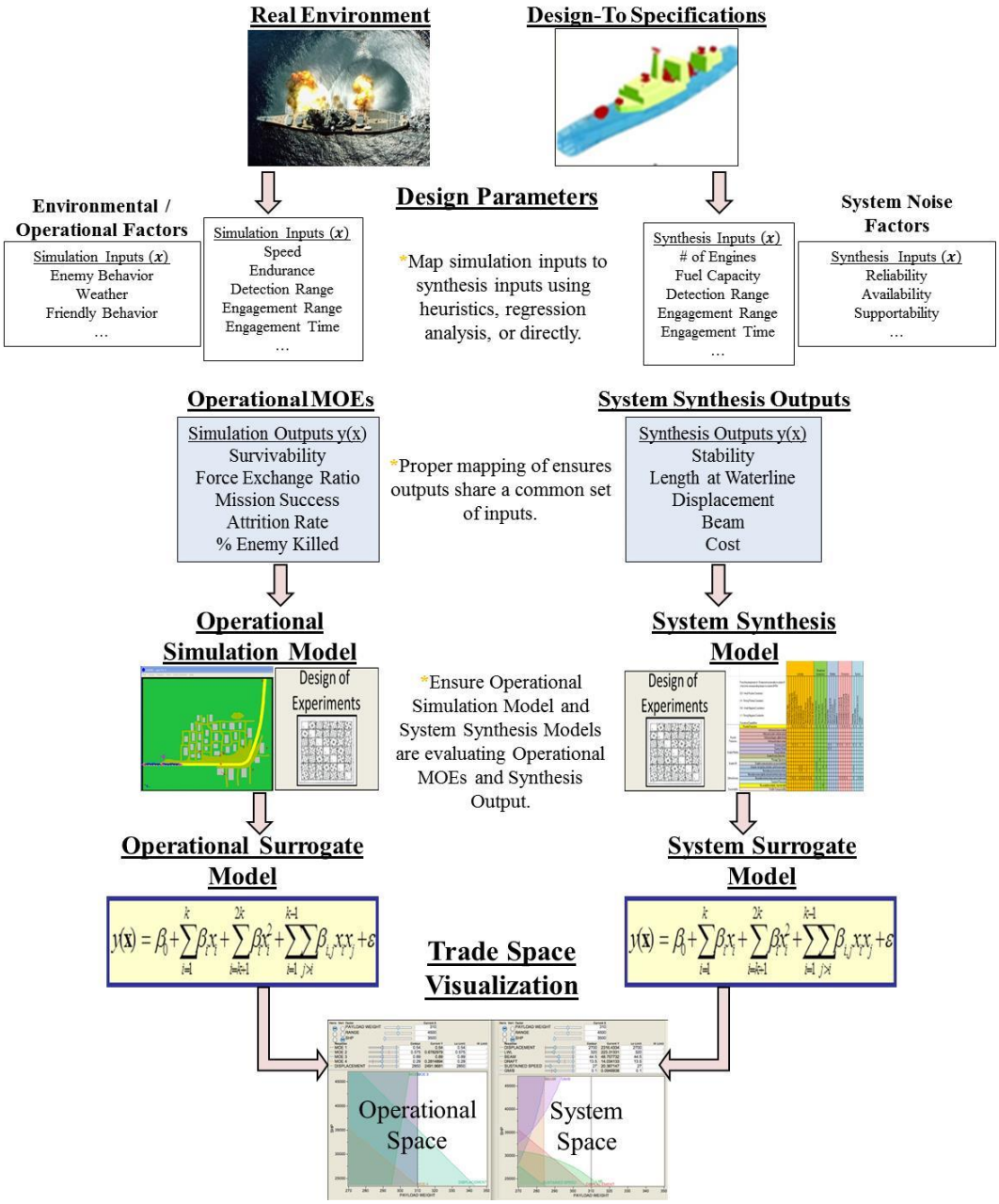


Figure 26 establishes a baseline analysis methodology implemented subsequent to the development of SysML products. Integration of this baseline analysis methodology with the SysML products outlined earlier establishes an MBSE MEASA. Five distinct stages comprise the MBSE MEASA and demonstrate that the MBSE MEASA conforms to the generic systems engineering process identified previously. Recall that the generic

systems engineering process takes a set of system requirements (in terms of SysML, these can be captured in a Requirement Diagram), identifies the functions that support those requirements (in terms of SysML, these can be captured in Activity, Sequence, Use Case, and State Machine Diagrams), identifies the physical elements that enable performance of those functions (in terms of SysML, these can be captured in Block Definition and Parametric Diagrams) and performs some analysis that can be used to assess how well those physical elements satisfy each function (and, by extension, how well a physical system satisfies identified requirements).

Figure 27 presents a visual construction of the MBSE MEASA. SysML modeling supports the first three stages of the methodology. Experimental design selection, simulation analysis, and trade space analysis support the final two stages. Figure 27 segments the MBSE MEASA into these five stages and identifies the SysML products and simulation analysis products that support each stage of the process. Note that the MBSE MEASA depends on generation of SysML products, but expands the scope of SysML modeling by adding the Analysis Methodology process (DOE Selection, Simulation Analysis, and Trade Space Analysis). Application of the MBSE MEASA ensures that SysML architecture products directly link to an analysis approach. This prevents development of overly complicated SysML architecture products (note that the MBSE MEASA also links these SysML products to traditional systems engineering product groupings) that remain stagnant and cannot be used to make actionable decisions. This also facilitates rapid iteration of the MBSE MEASA; Section C will discuss and demonstrate iteration in detail.

Figure 27 MBSE MEASA

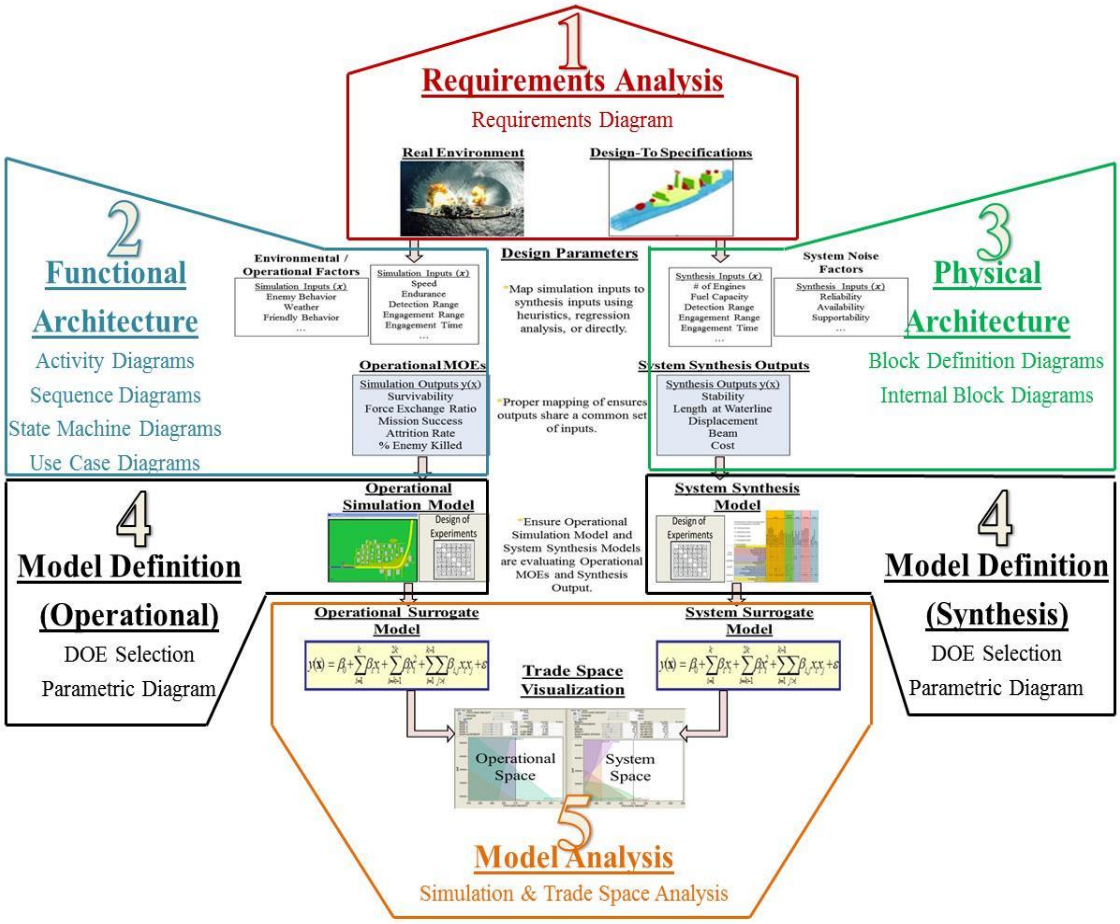


Figure 27 is information dense and may appear overly complicated; accordingly Figure 30, Figure 35, Figure 43, Figure 47, and Figure 49 segment the process and present the details associated with each step. However, an initial discussion of the overall goal of segmenting the analysis methodology is required prior to isolated discussion of each phase. The first goal of segmenting the MBSE MEASA definition of a process based on successfully generated SysML products. The second goal specification of the SysML products required to support each stage of the analysis process. The integration of SysML products as the enablers for the development of system architecture product fills the gap identified in Chapter I. Specifically; Step 1 develops a Requirement Diagram to capture both the environment and set of design specifications for the system, which aligns with the initial step of the generic systems engineering process. The MBSE MEASA

subsequently recommends the SysML products that define both the system functional architecture and physical architecture, as is recommended in the generic systems engineering process. The MBSE MEASA then uses those products to support development of external models and simulations, a vital expansion of the current MBSE methodological process and the primary enabler of the linkage between the system architecture and system analysis domains. This segmentation of the MBSE MEASA illustrates how each component of the MBSE MEASA supports creation of the previously identified products essential to realization of the generic systems engineering process. Table 1 provides a template that is updated throughout the dissertation to identify how each step of the MBSE MEASA supports creation of vital systems engineering products, defined earlier in this dissertation as: a Defined Problem, a Defined System Boundary, a Defined System Objective, Defined System Requirements, Defined Functional Behaviors, Defined Functional Performance, Defined Allocation of Requirements to Functions, Defined Candidate Physical Solutions, a Defined Model of Physical Solutions, Evaluation of Candidate Physical Solutions and an Assessment of Physical Solutions' Satisfaction of System Requirements.

Table 1 Template for Linkage of MBSE MEASA Steps to Systems Engineering Products

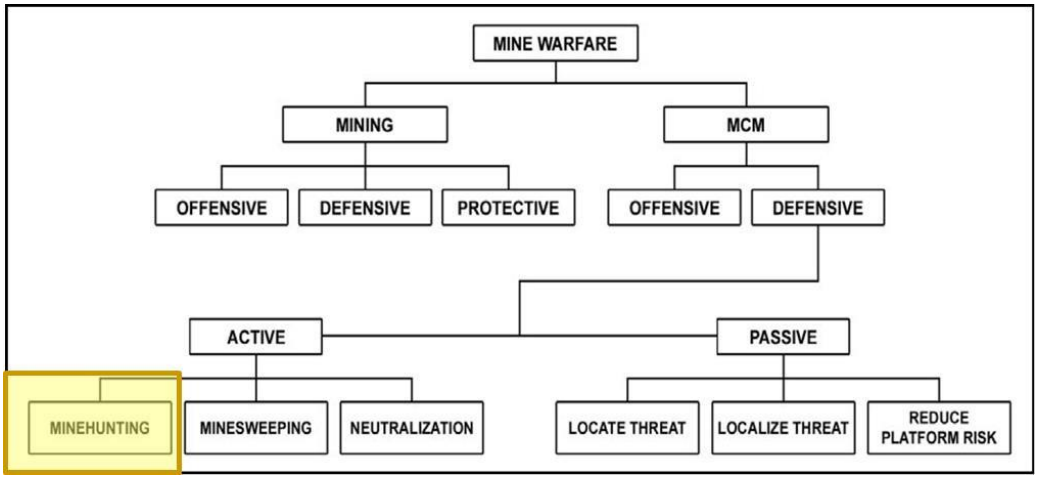
	Defined Problem	Defined System Boundary	Defined System Objectives	Defined System Requirements	Defined Functional Behaviors	Defined Functional Performance	Defined Allocation of Requirements to Functions	Defined Candidate Physical Solutions	Defined Model of Physical Solutions	Evaluation of Candidate Physical Solutions	Assessment of System Requirements
1. Requirements Analysis											
Requirements Diagram											
2. Functional Architecture											
Activity Diagrams											
Sequence Diagrams											
State Machine Diagrams											
Use Case Diagrams											
3. Physical Architecture											
Block Definition Diagram											
Internal Block Diagram											
4. Model Definition											
DOE Selection											
5. Model Analysis											
Simulation Analysis											
Trade Space Analysis											

3. Introduction to Mine Warfare Operations

This chapter focuses on presentation of the MBSE MEASA. Prior to presentation of the methodology, this chapter presents an example mine warfare (MIW) system that provides context for presentation of each step of the methodology. This research presents a demonstration of MBSE MEASA using the same mine warfare system in Chapter IV. This research builds off of the graduate research of Becker et al. (2014) which developed functional architecture (in the form of EFFBD) and physical architecture products that characterized the activities associated with mine warfare. That research developed a discrete event simulation model, which was analyzed to compare the effectiveness of the Littoral Combat Ship (LCS) and MCM-1 Avenger Class ship in Mine Countermeasure (MCM) operations. That same simulation model, with a few minor updates, is used in this dissertation.

A brief introduction to MIW operations is necessary to support understanding of each product developed in the MBSE MEASA prior to a detailed examination of each step of the MBSE MEASA. Carpenter (2010) provides an overview of the current and future challenges associated with MIW. Figure 28 illustrates the scope of MIW operations and also identifies the MIW operations of interest to this analysis.

Figure 28 MIW Activities



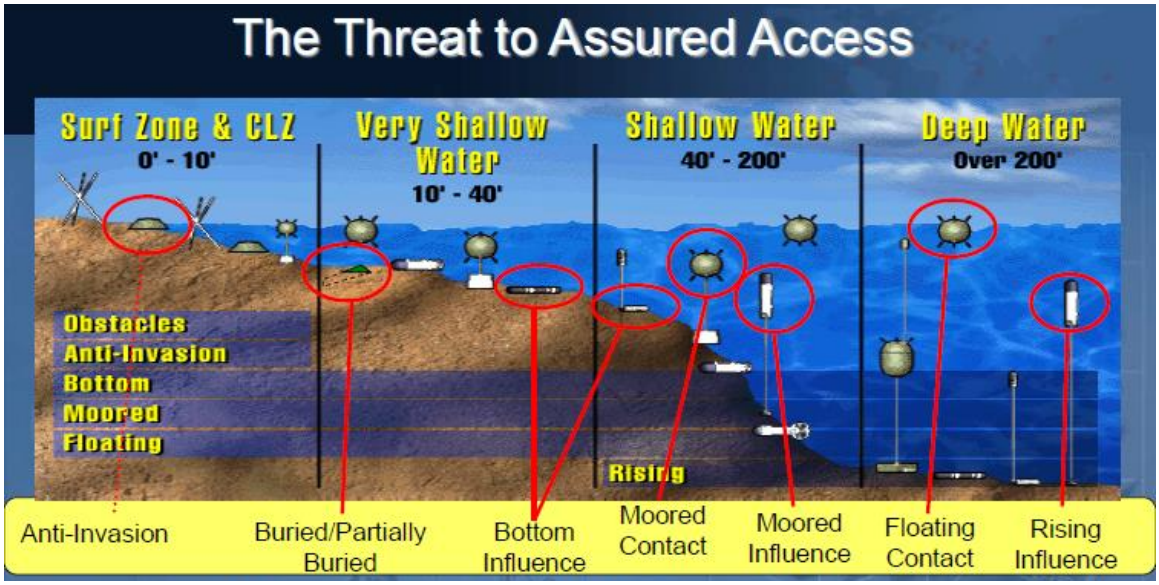
Adapted from: Carpenter, Wendi B. 2010. Navy Warfare Publication: Naval Mine Warfare. Vol. 1. NWP 3–15. Norfolk, VA: Navy Warfare Development Command.

MIW Operations are vital to the ability of a Navy to conduct uninhibited operations in strategic areas. Benes and Sandel (2009) show that, since 1950, mines damaged more U.S. Navy ships than missiles, torpedoes, aircraft, and small boats combined. Accordingly, the U.S. Navy has shifted resources toward MIW operations. Note that MIW encompasses both Mining Operations and MCM operations, two distinct challenges. This research facilitates comparison between legacy and future MCM operations by focusing on defensive MCM operations. Offensive MCM operations focus on neutralizing an enemy's ability to conduct mining activities, which is a challenge addressed by non-MIW dedicated assets. Furthermore, Active MCM operations are more relevant than Passive MCM operations, which are concerned with the ability to locate and avoid mines. Because Passive MCM operations are so dependent on operational decisions, they are not appropriate for consideration with the MBSE MEASA. Focus on

Active, Defensive MCM Operations bounds the overall MIW problem and focuses this research on a problem area that is well understood but requires additional investigation.

Further explanation of MCM Operations facilitates understanding of MCM systems. Sandel (2008) provides an overview of the activities typically performed in support of Active, Defensive MCM Operations. Those activities are: Detection, Classification, Identification, and Neutralization. Detection is the process of segmenting underwater clutter into Minelike Echoes (MILECs) and Non-Minelike Echoes (Non-MILECs). Classification is the process of classifying MILECs as either Minelike Contacts (MILCOs) or Non-Minelike Contacts (Non-MILCOs). Identification is the process of identifying MILCOs as Identified Mines and Identified Non-Mines. Neutralization is the process of successfully or unsuccessfully neutralizing Identified Mines. The details of each activity conducted within Defensive MCM operations supports development of architecture views later in this research. Additional clarification regarding the scope of this research is required before presenting those architecture views. Underwater mines can take several forms, but there are two types of mines (in terms of activation methods) of particular interest. Contact mines activate through contact with another object. Contact mines can fix to the seafloor, rest on the seafloor, bury, or float on the surface. Influence mines activate by either an acoustic, magnetic, pressure, or seismic signature. Like contact mines, influence mines can fix to the seafloor, rest on the seafloor, bury, or float on the surface. This research focuses on the ability of current systems to detect, classify, identify, and neutralize influence mines. Figure 29 provides a visualization of the types of mines of interest to Defensive MCM operations.

Figure 29 Types of Underwater Mines



Source: Amador, Brian. 2011. "U.S. Navy Funding Goals for Future Mine Warfare Capability." Lecture at the 16th Annual Expeditionary Warfare Conference, Panama City, FL.

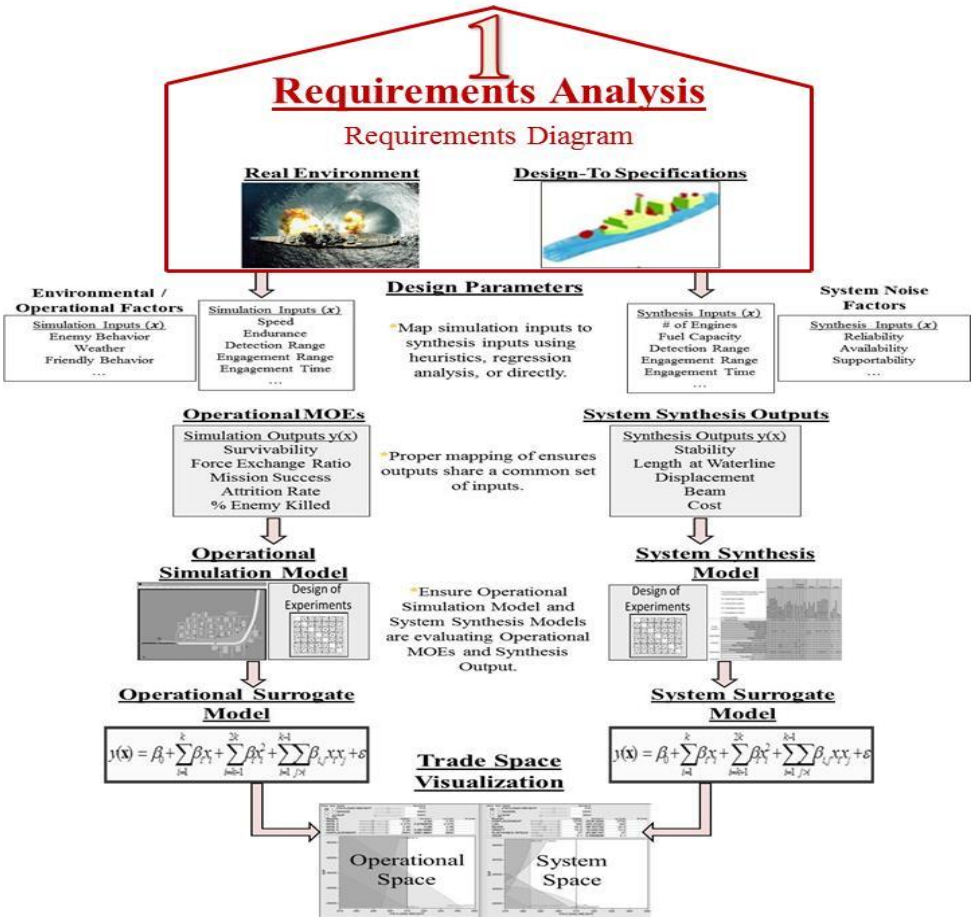
The intent of this review is not to provide a comprehensive overview of MIW operations, but rather to establish a focus area for the presentation of the MBSE MEASA. In particular, substantial development is necessary in each area of MIW Operations. This research uses Active, Defensive MCM Operations as a demonstration case. The above familiarization provides direction for the introduction to example system architecture views. Chapter IV provides more detail regarding Active, Defensive MCM operations for influence mines prior to demonstration of the MBSE MEASA.

4. Requirements Analysis Products

The goal of the stakeholder analysis phase, which is completed using a Requirement Diagram (and potentially a Package Diagram), is to summarize the system into a series of "The system shall" statements, both in terms of the operational environment and design specifications. This is supported by generation of a SysML Requirement Diagram (SysML Package Diagrams may also be used to support organization of system documentation). Figure 30 provides a visual representation of the first step in the MBSE MEASA, which uses a SysML Requirement Diagram to define a

set of system requirements that capture both the intended operational environment and design specifications for the system.

Figure 30 MBSE MEASA (Step 1)



Generation of the SysML Requirement Diagram for the MBSE MEASA assumes that a stakeholder analysis has been conducted and a set of requirements has been developed (this research does not assume that these requirements are necessarily "good" and investigation of the quality of those requirements is a major focus of the MBSE MEASA). Requirement Diagrams development is often the final step of the stakeholder analysis process, and is often supported through development of more traditional systems engineering products. In particular, visualization and communications of system

requirements is challenging, and is often accomplished through presentation, discussion, and iteration Integrated Definition (IDEF) models (in particular, IDEF0 models). Note that IDEF0 models are typically described as functional models; however the level of detail presented in these models as well as the comfortable description of the system in terms of inputs, outputs, controls, and mechanisms make them extremely useful for discussion with stakeholders who may not be familiar with the formal definitions associated with more detailed models. The National Institute for Standards and Technology (1993), advocate the use of IDEF0 models in this context and defines IDEF0 models as graphical system representations that describe the system in terms of the functions and activities that the system will perform, as well as the data, objects, and information that inter-relate the functions and activities.

Figure 31 provides an example of an IDEF0 model that was developed based on the information provided in Carpenter (2010) and Sandel (2008) to describe the general functions and activities of an active, defensive MCM operation. In general, such diagrams may be developed through analysis of supporting documentation or through interaction with project stakeholders. It is expected that most systems engineers will be familiar with the construct of IDEF0 models, the unfamiliar reader should refer to the original IDEF0 definitions presented in National Institute for Standards and Technology (1993). As a brief introduction, IDEF0 models present each function associated with a process in a box. These boxes accept inputs on the left and transform them into outputs on the right. Controls are shown at the top of each box. Controls present the conditions necessary for each function to take place. Mechanisms are shown at the bottom of each box. Mechanisms are the human or component resources necessary for each function to take place. Many approaches exist to guide the development of IDEF models. It is often easier to develop a conceptual understanding of the full system, as well as its interactions with external systems and the environment, by first considering the system as a subsystem within a larger context level IDEF representation. Figure 31 presents such an IDEF model, which presents the functions that define Defensive MCM Operations.

Figure 31 Context IDEF0 Model

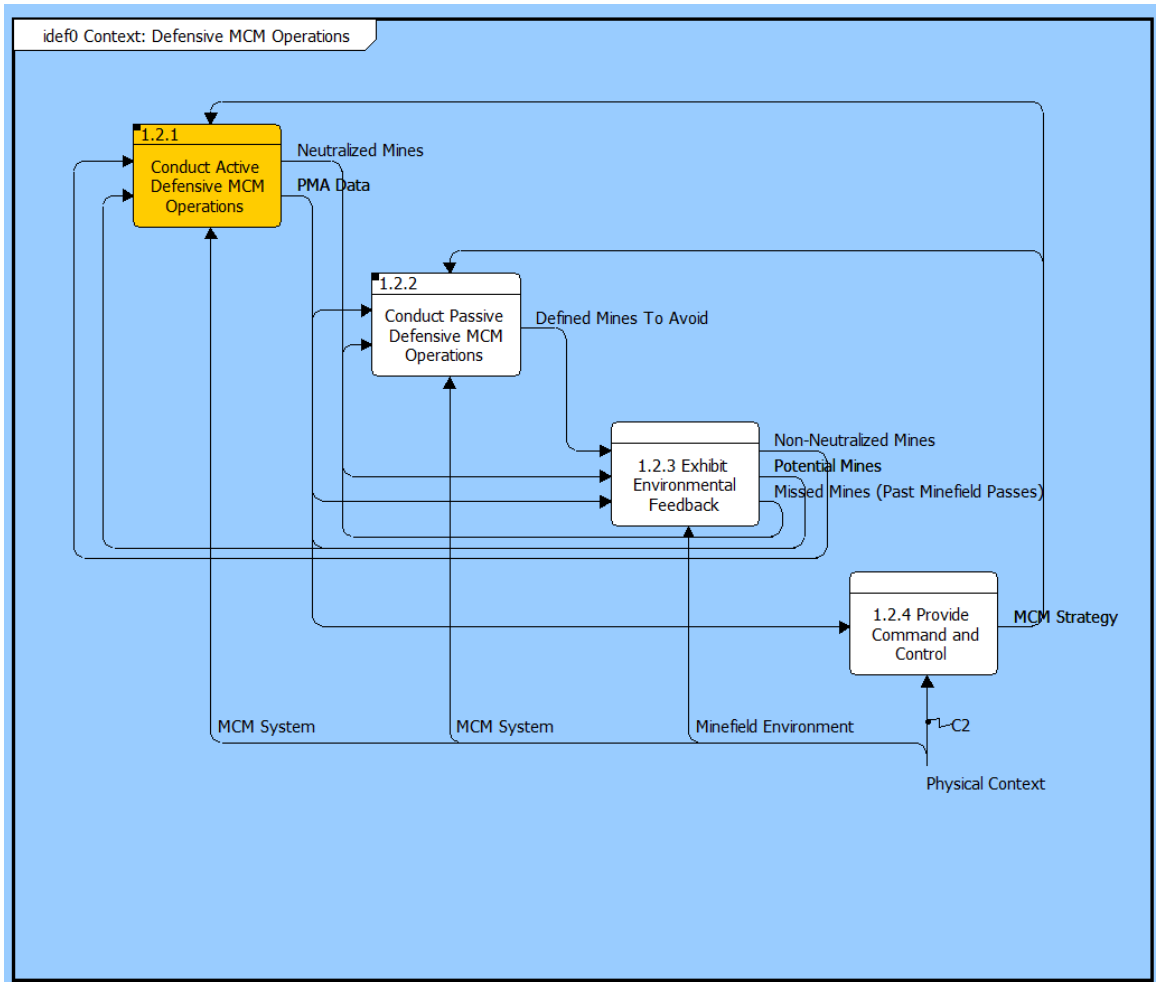
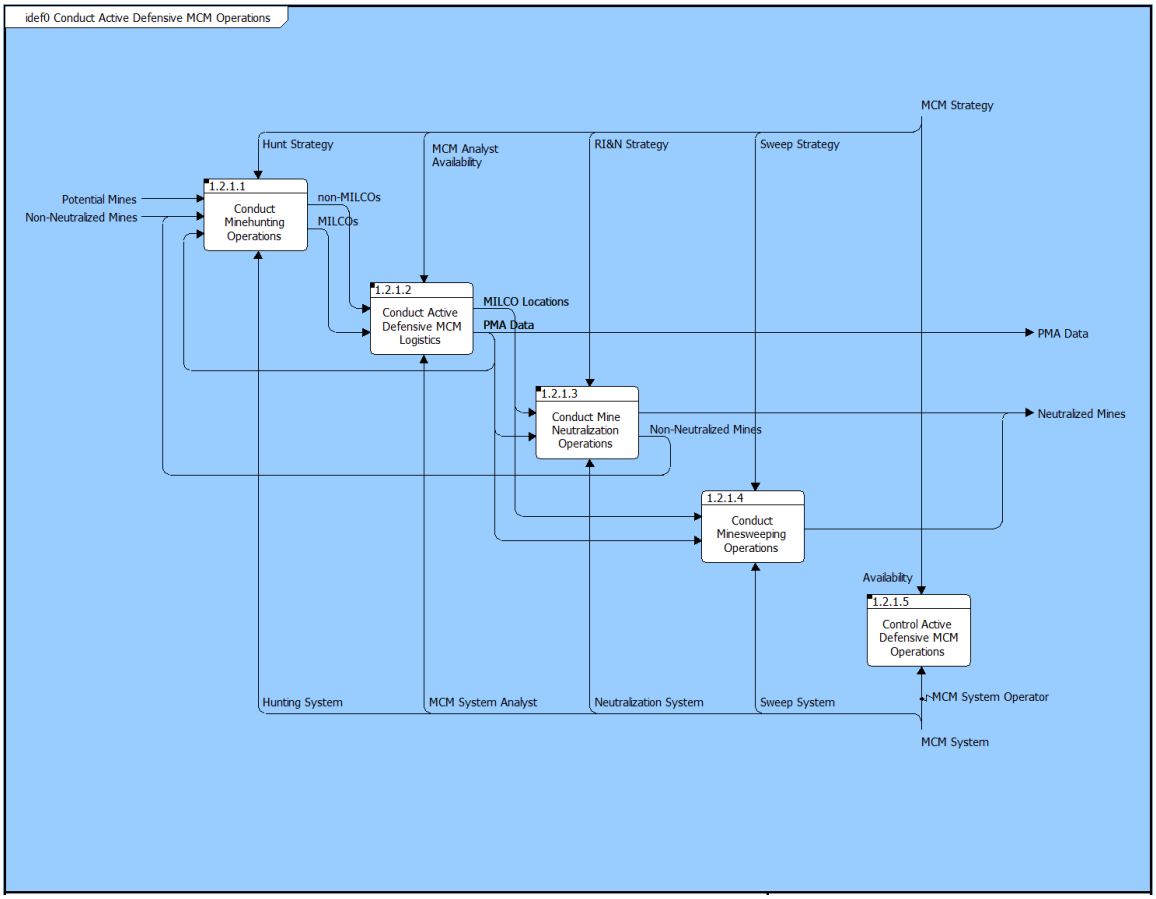


Figure 31 presents the Active, Defensive MCM Operations of interest to this research in the context of larger Defensive MCM Operations. Note that the Active, Defensive MCM Operations function (highlighted in gold for emphasis) accepts inputs (Potential Mines and Non-Neutralized Mines) from the Exhibit Environmental Feedback function. The Active, Defensive MCM Operations uses a generic MCM System (shown on the bottom as a mechanism), controlled by a generic MCM strategy (shown on the top as a control that is created by the Provide Command and Control function) to create both Neutralized Mines and Post Mission Analysis (PMA) Data. Both the Neutralized Mines and PMA Data are inputs to the Environmental Feedback function, which creates a list of Missed Mines (which is sent back to Passive Defensive MCM Operations) and Non-

Neutralized Mines, which prompts another instance of the Active, Defensive MCM Operations function. This diagram establishes a baseline understanding of the high level behaviors that must be represented in any system simulation. Specifically, a simulation of Active, Defensive MCM Operations must represent an MCM System that follows a set MCM Strategy, accepts a list of Potential Mines and Non-Neutralized Mines, and converts them to Neutralized Mines and creates PMA Data. Figure 32 presents a similar IDEF0 model for Active, Defensive MCM Operations and provides increased detail regarding the functions that define Active, Defensive MCM Operations.

Figure 32 IDEF0 Model for Active, Defensive MCM Operations



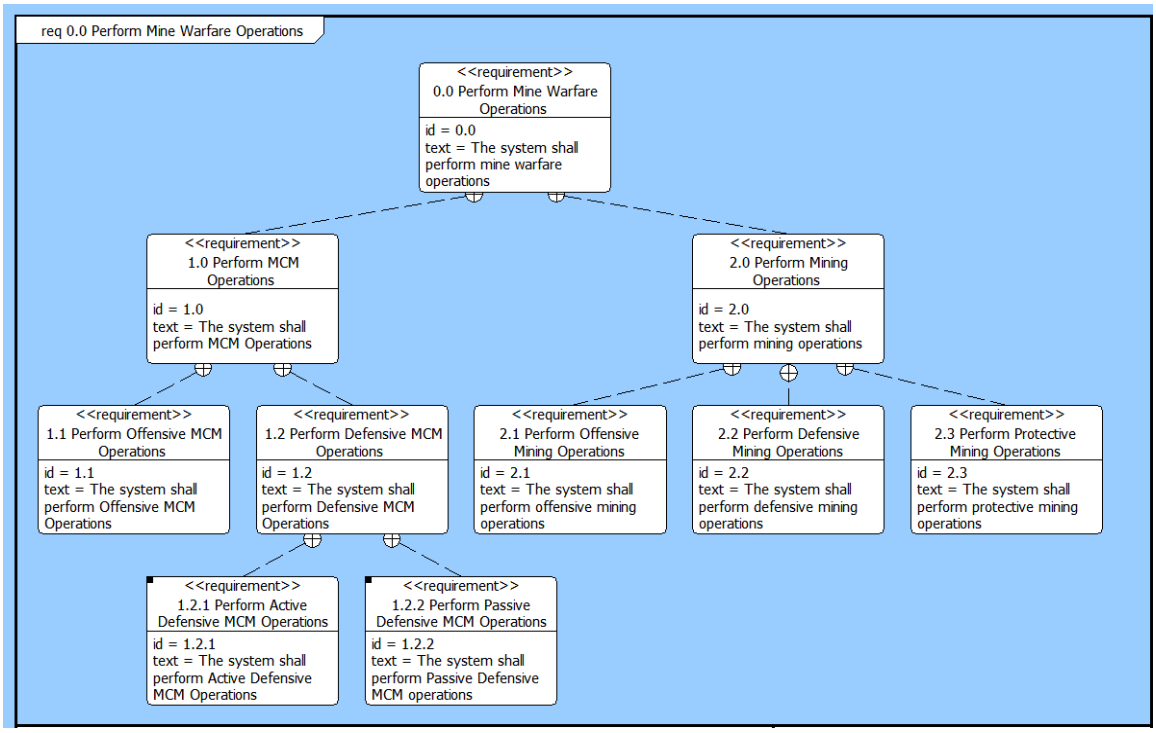
The IDEF0 model in Figure 32 presents the functions associated with Active, Defensive MCM Operations, as well as the inputs, controls, outputs, and mechanisms associated with each function. A brief examination of the inputs and outputs captures the

primary goal of Active, Defensive MCM Operations. The system accepts Potential Mines and Non-Neutralized Mines and outputs Neutralized Mines and PMA Data (the same set of inputs and outputs shown in the Context Level IDEF0). The expanded IDEF0 model for Active, Defensive MCM Operations details that conversion and can develop a more detailed understanding of the processes that must be represented in a simulation of Active, Defensive MCM Operations. This detailed examination of Figure 32 shows the humans and components essential to the process (each of which is a decomposition of the generalized MCM System previously shown in Figure 31) as well as the conditions necessary to conduct the process (each of which is a decomposition of the generalized MCM Strategy previously shown in Figure 31). Active, Defensive MCM Operations begin with Minehunting, which subsequently prompts MCM logistics functions, which in turn prompts Mine Neutralization and Minesweeping. The IDEF0 model also represents operational control. Each of these functions can also be decomposed to fully understand each sub-process associated with Active, Defensive MCM Operations. The IDEF0 captures the processes, system components, and conditions associated with Active, Defensive MCM Operations beyond the general description presented earlier. While this decomposition is extremely valuable for communications with stakeholders, the MBSE MEASA advocates the definition of system processes and components using SysML products, which are more easily translated to external models and simulations.

As mentioned, IDEF0 models facilitate easy communications with stakeholders, although perhaps more importantly they are a starting point for the generation of Requirements Analysis Products. Development of this diagram is vitally important because it ensures consistent terminology in both the functional and physical architectures (and later, between operational models and synthesis models). A high level SysML Requirement Diagram describes the general requirements for an Active, Defensive MCM system. The general requirement “Perform MCM Operations” aggregates lower level requirements. A SysML Requirement Diagram specifies any and all system requirements, including intended capabilities, expected functions, and performance conditions. Many requirements will describe an intended capability in terms of its expected functionality and quantify a performance metric. The SysML Requirement

Diagram provides that functionality and also allows for specification of relationships between requirements, as well as between requirements and other model elements. The types of relationships allowed within a SysML Requirement Diagram are: Requirements is satisfied by, Requirement is derived from, Requirement derives, Requirement is refined by, and Requirement is verified by. Figure 33 presents an example of a SysML Requirement Diagram for the Requirement "Perform Mine Warfare Operations."

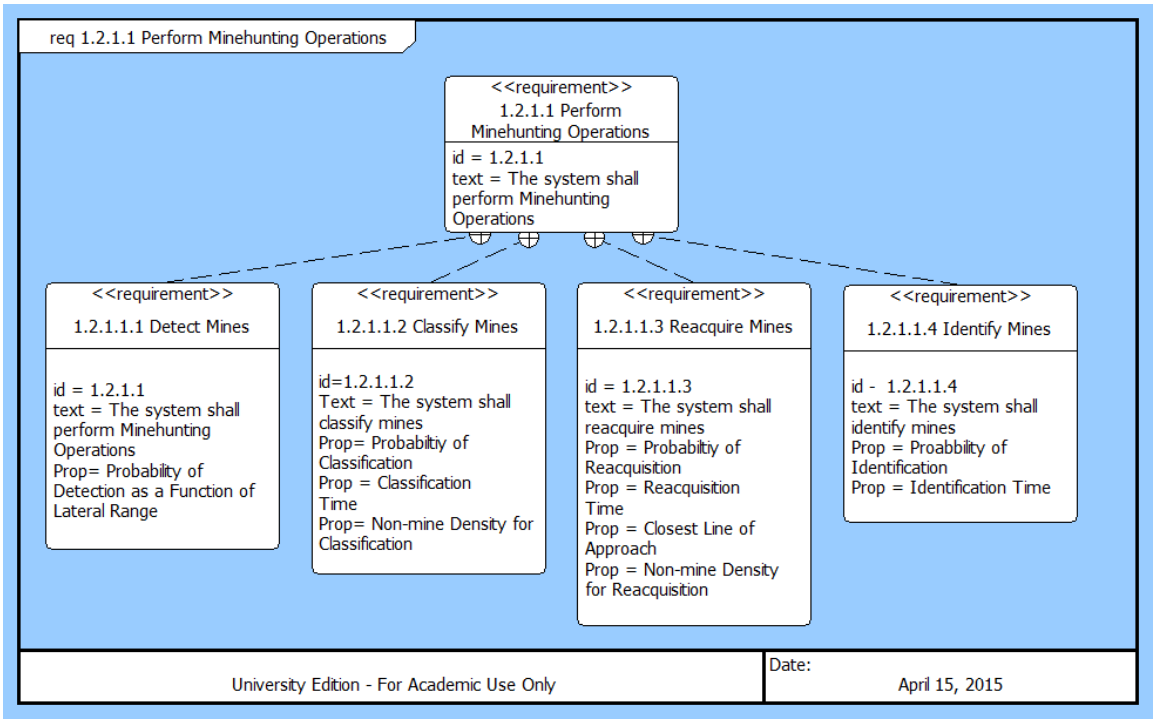
Figure 33 Requirement Diagram: Perform Mine Warfare Operations



While the high level SysML Requirement Diagram presented in Figure 32 does not capture all of the aspects of MCM operations, it does establish a common operating model that can be supplemented with increased detail. Figure 33 provides an example of an abstract, high level requirement (Perform Mine Warfare Operations), which is refined by additional requirements (ex: Perform MCM Operations). Figure 33 also shows several more detailed relationships (ex: Perform Defensive MCM Operations refines Perform MCM Operations).

For clarity regarding the level of potential detail that may be included in a Requirement Diagram, Figure 34 presents a decomposition of the Perform Minehunting Operations requirement, which is a requirement that refines Perform Active, Defensive MCM Operations, as shown in Figure 33. Note that the use of a consistent numbering convention clarifies the relationships between requirements; Detect Mines (requirement id 1.2.1.1.1) refines Perform Minehunting Operations (requirement id 1.2.1.1), which refines Perform Active, Defensive MCM Operations (requirement id 1.2.1). In terms of requirement development in support of the MBSE MEASA, which necessarily relies on Requirement Diagrams for definition of system level performance parameters as well as guidance regarding development of operational MOEs, several specific steps should be taken when developing Requirement Diagrams. At a minimum, the MBSE MEASA recommends that all requirements include an id number as well as a text description. Requirements that are not refined by any additional requirements should also include a property, which should be a quantifiable, assessable quantity. Assessment of these quantifiable metrics for lower level requirements provides assessment of the high level requirements (even those that are text based). This defines the system parameters, environmental factors, and operational factors represented in an external model, and establishes traceability between those performance parameters and operational MOEs. While each requirement can be supplemented with additional detail regarding the criticality or risk of the requirement, it is recommended that these characteristics not be ascribed to any requirements prior to initial examination of an external model, which will provide insight regarding the impact of each system design parameter on the performance of the overall system. If a stakeholder identifies a requirement as critical this should be included in the text description of the requirement for examination in future iterations of the MBSE MEASA.

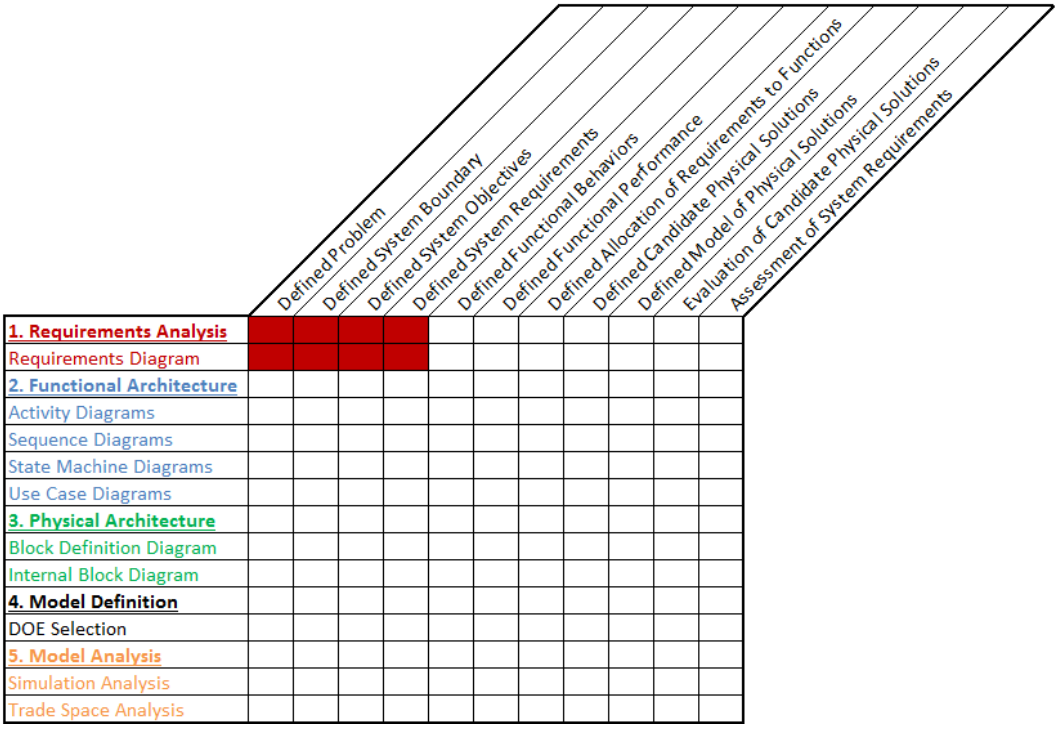
Figure 34 Requirement Diagram: Perform Minehunting Operations



Note that each requirement that is not refined by additional requirements includes at least one property that can assess the system performance in terms of that requirement. As mentioned, this is vitally important because these properties will guide the selection of input variables to all external simulation models. Note that, as the MBSE MEASA is iterated, these properties may be supplemented with increased detail. As an example, Figure 34 presents Requirement 1.2.1.1.4 (Identify Mines) on the far right. Currently, two properties define Identify Mines: Probability of Identification and Identification Time. During a subsequent iteration of the MBSE MEASA (which should be based on the analysis of a system simulation model) this property may be specified with more detail (ex: Probability of Identification greater than 0.80). The level of detail shown in Figure 34 presents an expected level of definition for each higher level requirement shown in Figure 33. The first step of the MBSE MEASA (Requirements Analysis) is considered complete after a Requirement Diagram captures the full set of stakeholder needs and a quantifiable metric is established for each low level (unrefined) requirement. A completed Requirement Diagram should include sufficient detail to accomplish the first

four systems engineering products presented earlier. Table 2 presents an updated linkage of the systems engineering products supported after Step 1 (Requirements Analysis) of the MBSE MEASA.

Table 2 Requirements Analysis Support of Linkage of MBSE MEASA Steps to Systems Engineering Products



Development of a comprehensive Requirement Diagram allows any user to communicate a clearly defined problem system boundary, system objectives, and system requirements. This aligns with the previously presented definition of system requirements as a set of "The system shall," statements that capture the operational environment and design specifications for the system in terms of intended capabilities, expected functions, and quantified performance conditions. Note that development of a SysML Package Diagram can be conducted prior to development of a Requirement Diagram, but is not necessary. The Package Diagram can organize the information collected from a Stakeholder Analysis. For example, a SysML Package Diagram can classify and organize system requirements, use cases, behaviors, structure, and definitions in a SysML

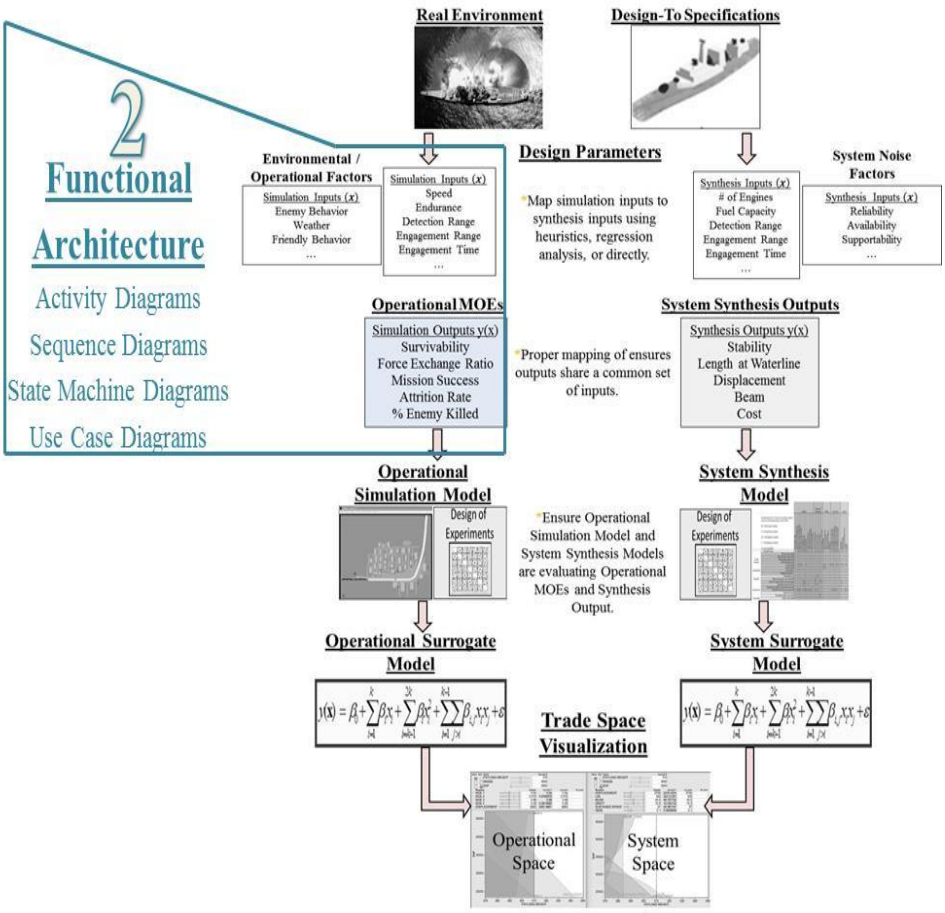
modeling tool. Using Package Diagrams as brainstorming organizational tools may ease the construction of future diagrams and products, but is not necessary, and therefore a detailed discussion is not included in this dissertation. For an in depth presentation of the potential value of beginning SysML modeling with the creation of a Package Diagram see Friedenthal, Moore, and Steiner (2009).

5. Functional Architecture Products

The system development process moves from Problem Definition to System Design after a Requirement Diagram is complete. System Design is defined as: Functional Architecture Development, Physical Architecture Development, Feasible Design Generation, and Modeling and Simulation. Note that in this context the term Modeling and Simulation describes the process of evaluating the ability of a given Physical Architecture to satisfy the functions outlined in a given Functional Architecture. Subsequent to the creation of a comprehensive Requirement Diagram SysML Diagrams capture these Functional Architecture products.

Functional Architectures summarize the system in terms of HOW it will satisfy the requirements identified in Step 1 (Requirements Definition), but do necessarily not define what physical system elements will satisfy those requirements. Development of SysML Activity, Sequence, State Machine, and Use Case Diagrams (Figure 35) support this definition. Further, the definition of the set of sequenced activities, the state dependent transitions between those activities, and the users responsible for the execution of each of those activities guides development of external operational simulation models.

Figure 35 MBSE MEASA (Step 2)

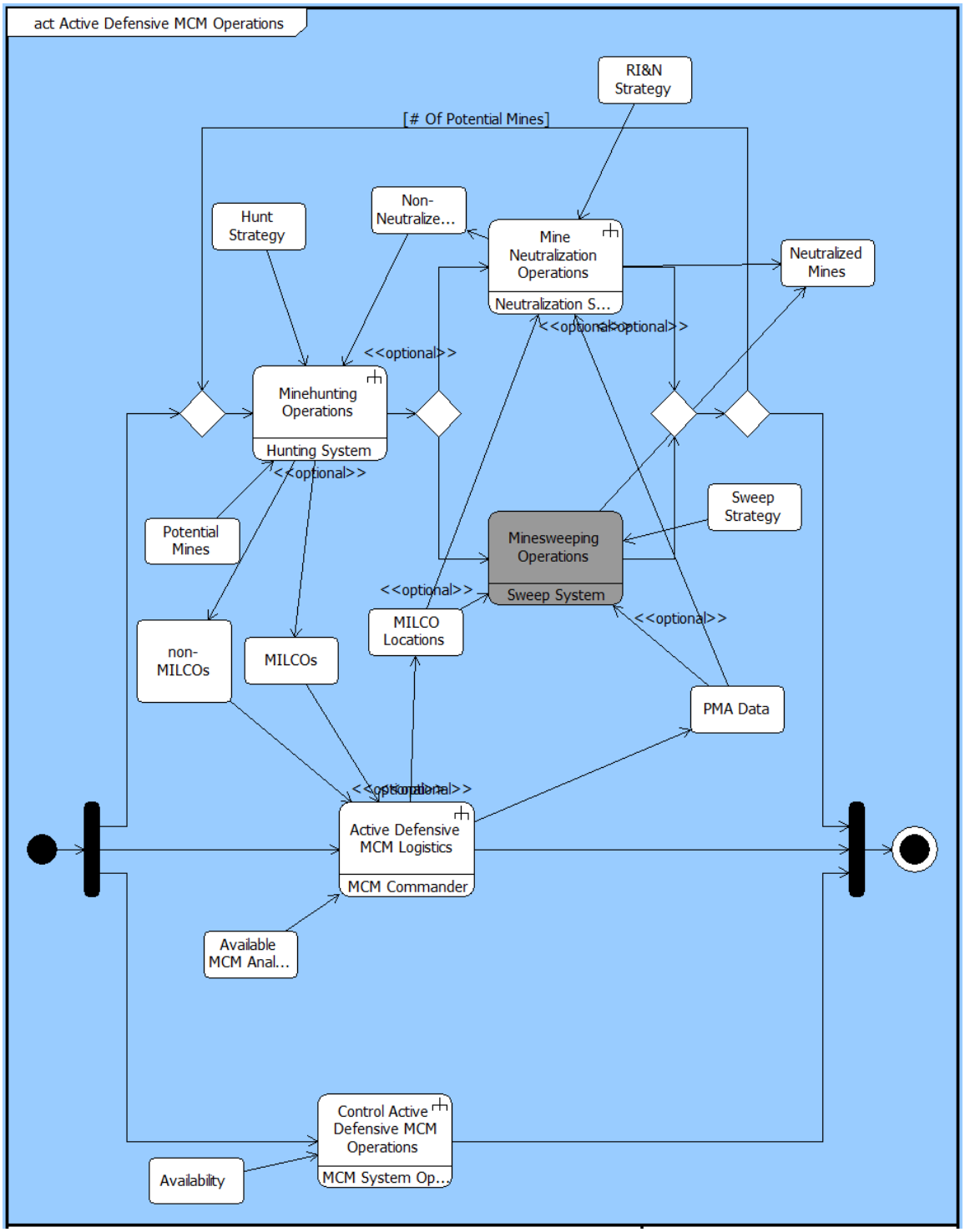


As mentioned, Functional Architectures specify how a system will behave. Accordingly, an Activity Diagram, which specifies what a system must do in order to satisfy requirements, is an appropriate first product to generate as part of a functional architecture. As mentioned when presenting the intended benefits of MBSE, the utilization of MBSE software to create various architecture views (rather than the utilization of static, standalone documents) ensures that conflicts are resolved between different types of diagrams as well as between different levels of each diagram. As with the IDEF0 models presented previously, it is difficult to capture the utility of this enforced consistency through the presentation of static figures. However, a major focus of this chapter is demonstrating the fundamental elements that can be included in each SysML Diagram, demonstrating the traceability and consistency that these diagrams

ensure, and highlighting the expected utilization of each diagram as a guideline for the development of external simulation models. Accordingly, this chapter will present several representations of each diagram, beginning with the simplest, highest level diagram of interest to this research (typically Active, Defensive MCM Operations), then moving to a decomposition of Minehunting Operations, and finally decomposing Mine Detection. Additional detail regarding Mine Neutralization, Mine Classification, Mine Reacquisition, Mine Identification, and MCM Logistics Functions will be included in Chapter IV as needed to demonstrate proper development of an external simulation models.

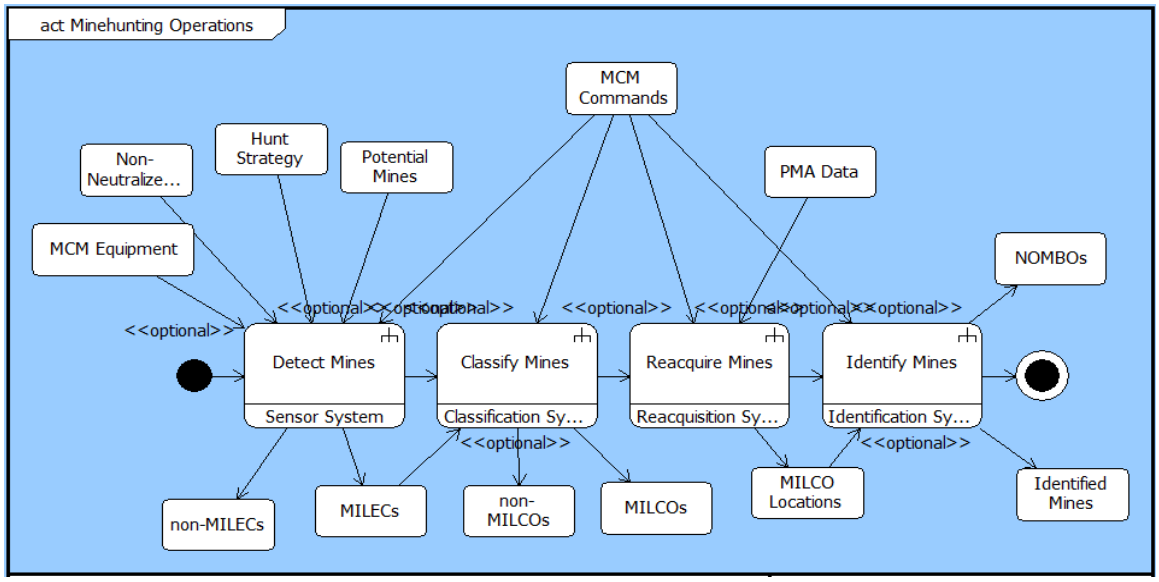
Activity Diagrams are a reasonable starting point for the development of functional architecture products for several reasons. Note only do Activity Diagrams describe what the system must do to satisfy each function, they also describe the external objects that are necessary to complete or trigger each function. Activity Diagrams can also model parallel operations, loops, iterations, and replications of activities. Also notable is the ability to group activities into partitions (also called swim lanes) that allow a user to specify responsibility (in terms of model parameters) regarding execution of those activities. While most activities must be completed to trigged subsequent activities, some activities within these partitions can be specified as interruptible if any stop or delay in that activity does not impact any other actions or activities. While these characteristics are difficult to understand through narrative text, they are easily visualized and understood through examination of the diagrams. As mentioned, the sequenced presentation of figures in this chapter demonstrates the fundamental elements and structure of each diagram in detail (to include highlighting traceability and consistency between diagrams) and demonstrates their utility in development of external simulation models. Figure 36 presents an Activity Diagram for Active, Defensive MCM Operations. Note that it provides similar detail to the IDEF0 model of Active, Defensive MCM Operations but also provides increased information regarding the ordering of each activity.

Figure 36 Activity Diagram: Active, Defensive MCM Operations



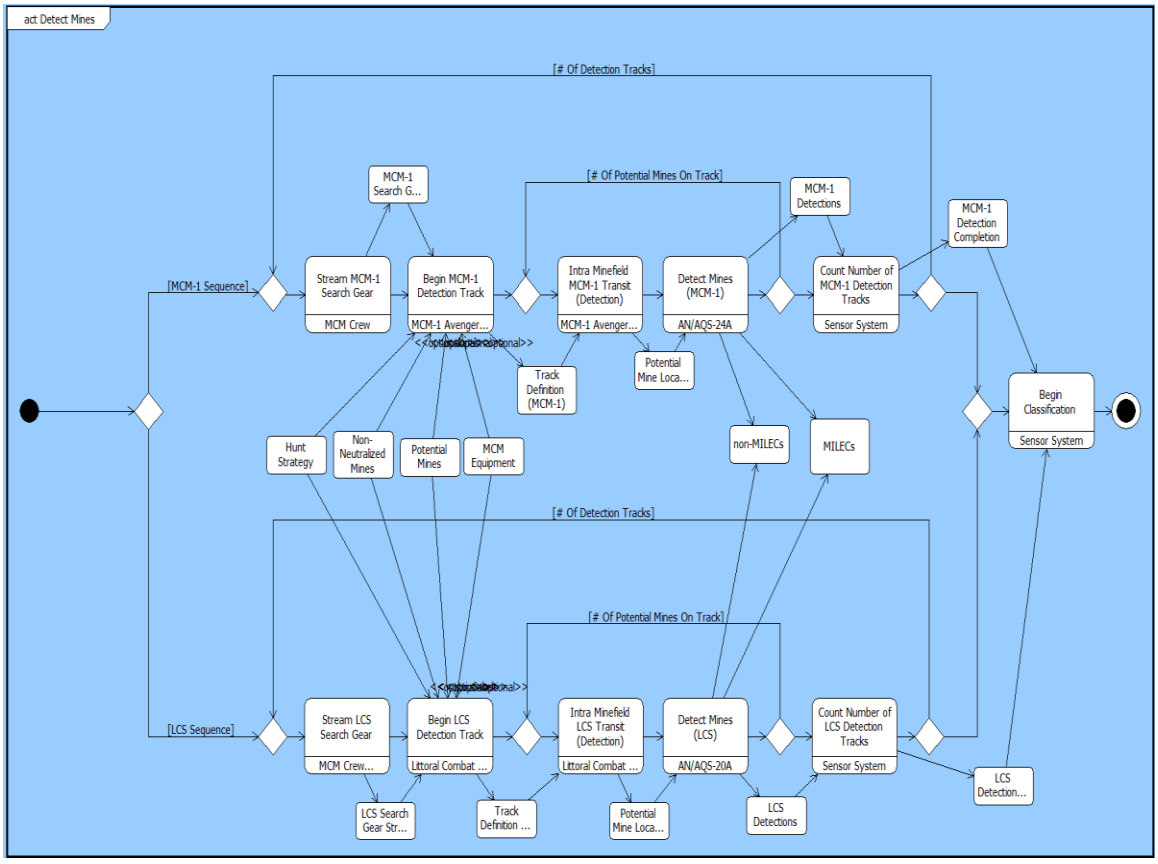
The Activity Diagram shown in Figure 36 demonstrates that there are three parallel processes associated with Active, Defensive MCM Operations: a sequence of Minehunting, Mine Neutralization, and Minesweeping; Active Defensive MCM Logistics; and Control of Active, Defensive MCM Operations. The parallel nature of these activities suggests that it may be necessary to concurrently allow for each of them within an external simulation model. The Activity Diagram also specifies that after Minehunting Operations conclude there is a choice between Mine Neutralization Operations and Minesweeping Operations. This research will focus on Mine Neutralization Operations (Minesweeping Operations has accordingly been shaded in gray by the author). This choice was made to facilitate a more accurate comparison between legacy and future MCM systems since there is very little detail available regarding the performance of the future minesweeping system that will be utilized by the LCS MCM Package. Note that the Activity Diagram also specifies the external components that will be created and used by each activity, which provides detail regarding the physical entities that must be represented in an external simulation model. It is possible to further decompose the Activity Diagram shown in Figure 36. Figure 37 presents an example Activity Diagram that describes Minehunting Operations.

Figure 37 Activity Diagram: Minehunting Operations



A straightforward sequence of events defines Minehunting Operations and therefore the Activity Diagram is actually simpler than the Activity Diagram for Active, Defensive MCM Operations. Minehunting Operations is comprised of: Detect Mines, Classify Mines, Reacquire Mines, and Identify Mines. Note that the Activity Diagram specifies the inputs that are necessary for each Activity, as well as the outputs that result from each Activity (some of which are subsequently used as inputs to other Activities). Because MILCOs, non-MILCOs, etc., may not exist in every scenario the Activity Diagram classifies some outputs as optional to indicate that their creation is not required to continue through the activity. Figure 38 presents a further decomposition of the Detect Mines activity and provides greater detail regarding the level of detail that Activity Diagrams can present.

Figure 38 Activity Diagram: Detect Mines

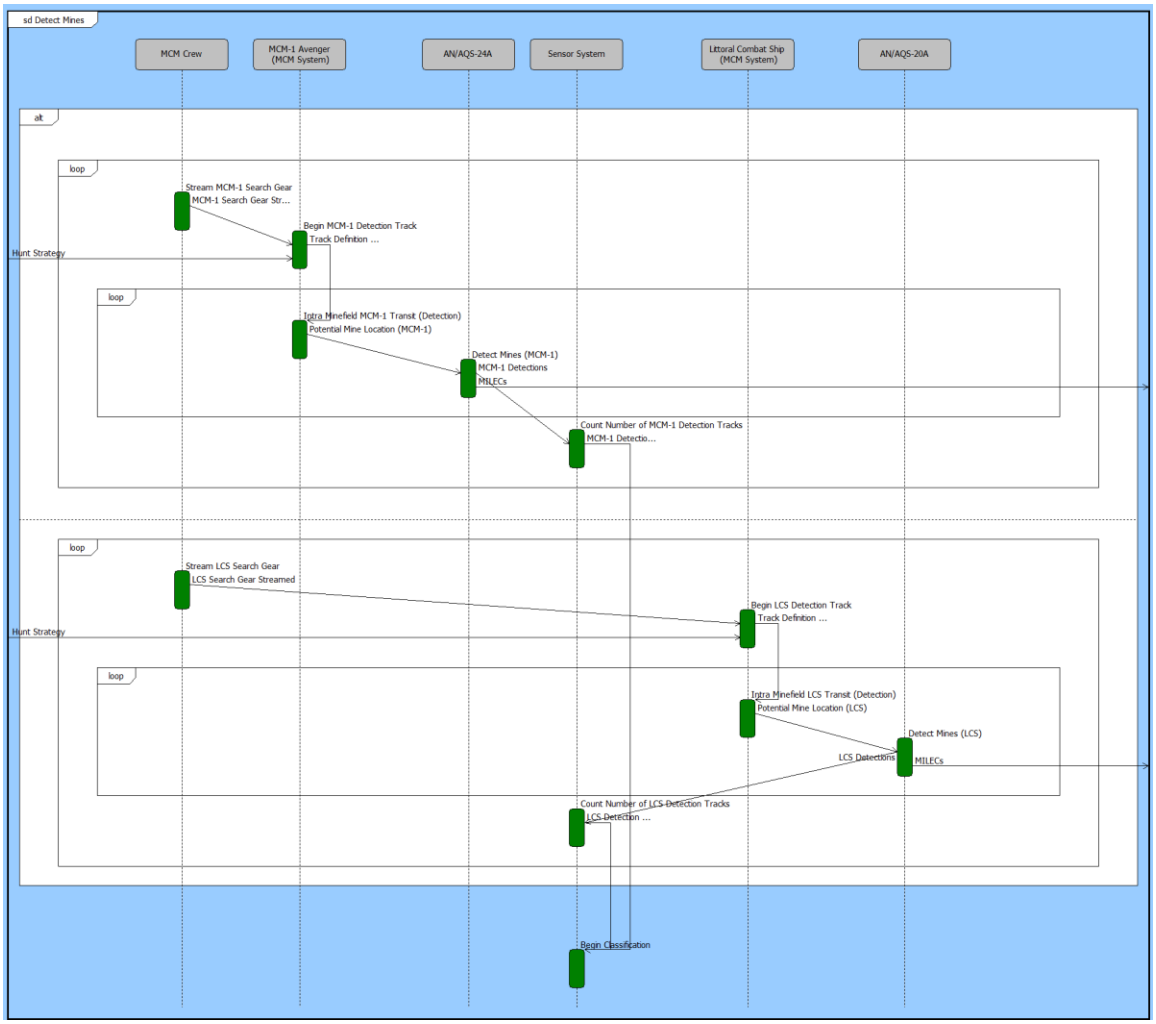


The Activity Diagram in Figure 38 shows that Mine Detection is initiated by the selection of either the MCM-1 Sequence (top) or the LCS Sequence (bottom). Each sequence then enters a loop that continues for a specified number of detection tracks. This loop begins with streaming of search gear and requires a hunt strategy, previously non-neutralized mines, other potential mines, and MCM equipment as inputs. The system then transits within the minefield and detects mines (that sequence is conducted for the number of potential mines on the track). That creates a list of MILECs and non-MILECs and the system then records the number of detection tracks, which may prompt an additional pass through the loop. If the loop has finished, the system begins classification, which uses the MILECs as an input. This sequencing is used to guide development of an external simulation model for Active, Defensive MCM Operations (in conjunction with functional architecture products that describe Mine Classification, Mine Reacquisition, Mine Identification, Mine Neutralization, and Control and Logistics of Active Defensive MCM Operations, which will be shown in detail in Chapter IV). The richness of Activity Diagrams, and their ability to represent behaviors through a presentation of not just activities (which translate nicely to events that must be modeled within external models) but also interactions and triggers (which translate generally to event sequencing and provide generic guidance for the development of physical architecture products) that are consistent across multiple levels of decomposition make Activity Diagrams a reasonable starting point for the development of functional architecture products in support of the MBSE MEASA. Note that the implementation of Activity Diagrams in the software program chosen by the author (CORE) can result in diagrams that are extremely busy. This is a result of a requirement within the software program that each diagram be "connected," meaning that any entity created within an activity be consumed and used within that activity. This can create extremely busy diagrams and cause issues when executing the architecture. Appendix C provides more details and recreates the architecture in an alternative software program that overcomes some of the limitations associated with this implementation of Activity Diagrams using CORE.

After an Activity Diagram describes all the activities that a system will complete, Sequence Diagrams provide additional information regarding interactions between elements of the internal structure of an activity. Generally, Sequence Diagrams supplement the information shown in Activity Diagrams by providing details regarding

what is necessary to support a particular activity, which helps provide clarity regarding ordering of activities. Specifically, it should alert any user to conflicts that may result from expecting an activity to commence prior to creation of external information necessary to support that activity, a level of detail that may be difficult to attain when using only Activity Diagrams, which provide no detail regarding the control of activity inputs or outputs while modeling at the level of abstraction shown. Figure 39 and Figure 40 present Sequence Diagrams for Mine Detection and Mine Classification that demonstrate the dependencies between the activities and within the sub-activities for each function, as well as the physical components responsible for the control of each activity.

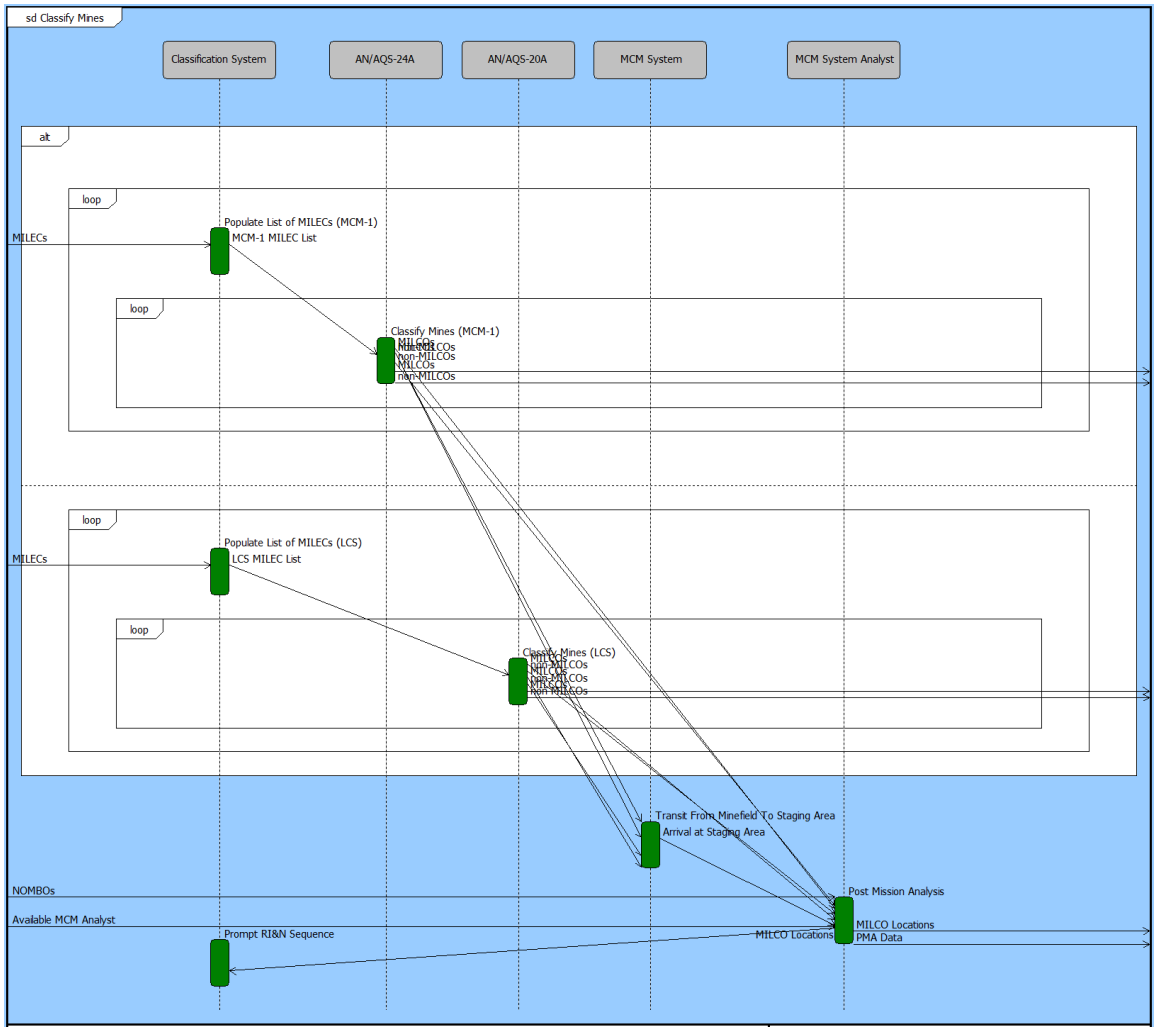
Figure 39 Sequence Diagram: Detect Mines



The Sequence Diagram for Mine Detection shown in Figure 39 provides additional clarity regarding Mine Detection beyond the information presented in the Activity Diagram for Mine Detection (shown previously as Figure 38). As with the Activity Diagram, there are two alternate series, one for the MCM-1 and one for the LCS. Each series is comprised of the same set of activities as shown in Figure 38, each series is conducted by distinct physical components (the physical components associated with mine detection are shown in gray boxes along the top of Figure 39 and vertical lines descending from each figure intersect with the activities that each physical component conducts). This formal definition of the physical components associated with each activity may guide the definition of resource requirements in an external simulation model and may aid in identifying potential conflicts when multiple events are dependent on a single physical component. Just as importantly, Sequence Diagrams trace not only events (as was shown in the Activity Diagram) but also of the outputs of each function and the triggers to each function. As a side note, Sequence Diagrams may be assessed for correctness by conducting a flow continuity check, which checks that the flow from the first activity to the final activity is possible without referencing activities shown in another diagram. Notice that the Sequence Diagram shown in Figure 40 would fail such a test, since MILECs are outputs of Mine Detection and leave the page on the right side. This is a function of the segmentation of Minehunting by the author to ease understanding. Specifically, the MILECs that exit the Mine Detection Sequence Diagram are immediately accepted by the Mine Classification Diagram (Figure 40). If Mine Detection and Classification were grouped as a single function and a combined Sequence Diagram was developed, the resulting Mine Detection & Classification Sequence Diagram would pass the flow continuity check for consistency. As mentioned, this segmentation was done to ease communication for the reader unfamiliar with MIW and MCM operations since static architecture figures that consider the entire sequence of mine detection through neutralization would be difficult to present. That said this visual issue with Figure 39 further demonstrates the value of Sequence Diagrams. They ensure that the outputs of each activity are used by a follow on activity, and identify potential

issues resulting from redundant activities, physical components that are simultaneously utilized by multiple activities, and dependencies between activities.

Figure 40 Sequence Diagram: Classify Mines



While Sequence Diagrams provide increased detail regarding system functionality, they are often focused on sequences of message exchanges from a control perspective and may not allow for maximum detail regarding specific actors in specific scenarios. Use Case Diagrams can be used to further aid development of functional architecture views by providing that increased level of detail regarding the actors that are involved in each activity. Use Case Diagrams are particularly useful for multi-purpose

systems, which may require a different set of personnel to execute each activity. Used in conjunction with Sequence Diagrams, this allows a systems architect to identify potential conflicts in terms of both system control and system implementation. Figure 41 provides an example of a Use Case Diagram.

Figure 41 Use Case Diagram: Perform Mine Hunting Operations

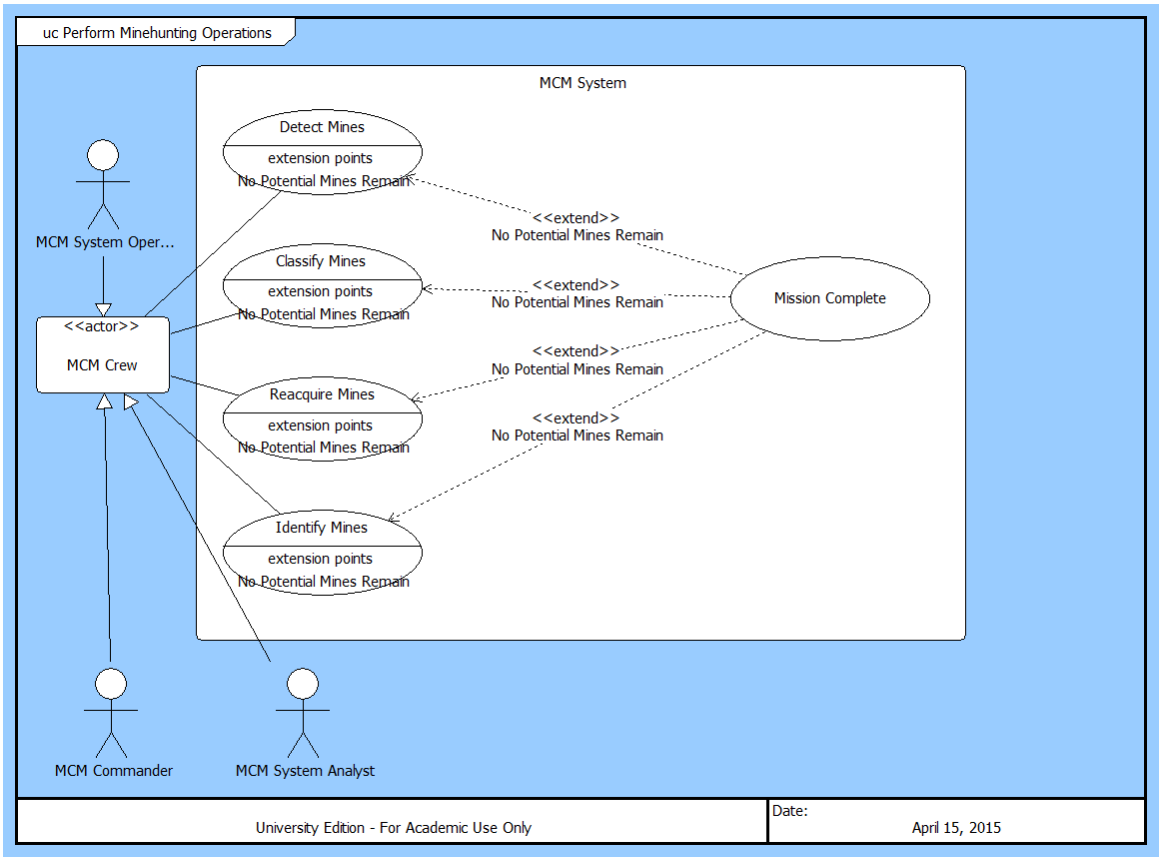
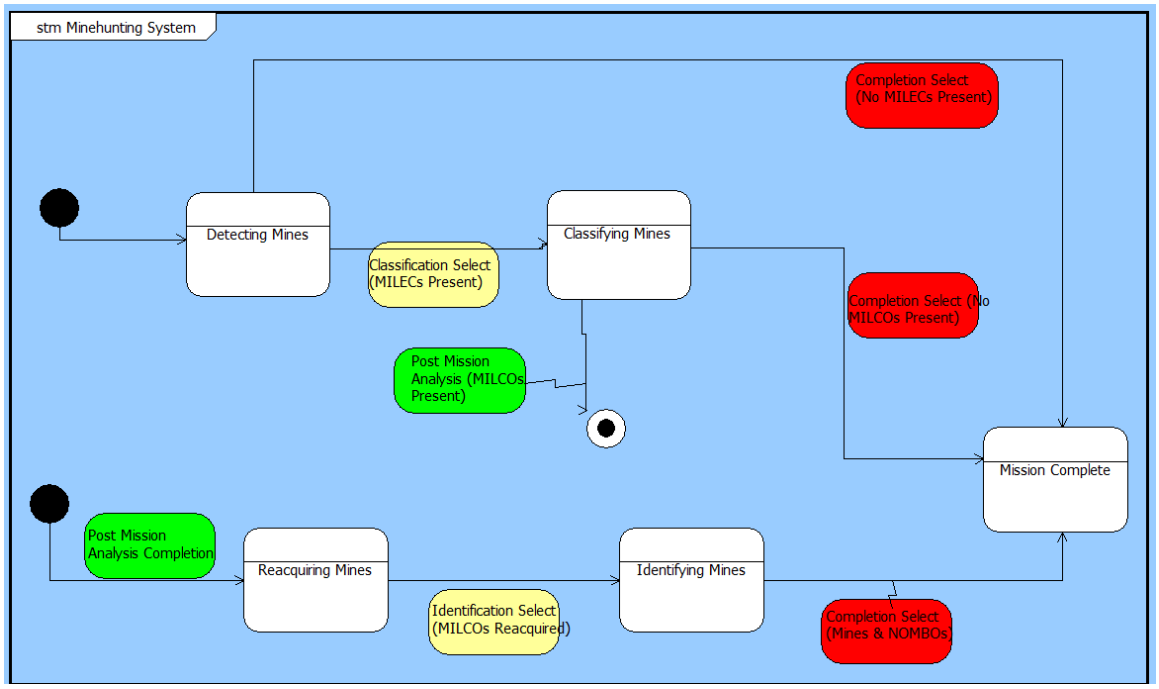


Figure 41 shows that the MCM System performs the Perform Minehunting Operations activity. As a point of emphasis, during the conceptual design phase all effort should be made to remain solution neutral, the Use Case Diagram uses the generic term "MCM System" rather than specifying that the system as MCM-1 Avenger or the LCS. The Use Case Diagram prescribes the same list of activities that define Minehunting Operations in the Activity Diagram in Figure 37 (Detection, Classification, Reacquisition, and Identification); however the Use Case Diagram also defines the actors

(MCM System Operator, MCM Commander, and MCM System Analyst, who are involved in each of these activities). Note that each of the actors is external to the system of interest and each actor participates in each activity. In this example each actor is also defined as a subclass of "MCM Crew," however this aggregation may not be appropriate in all cases. The Use Case Diagram also specifies that Mission Complete is a potential extension of each of the Minehunting Operations activities, which is triggered by the extension point "No Potential Mines Remain."

Development of an Activity Diagram prescribes the general functions that a system will perform as well as the outputs and inputs of each of those functions. Development of a Sequence Diagrams prescribes the ordering of those functions and also defines the environmental triggers that may be necessary to initiate each function. Development of a Use Case Diagram provides clarity regarding the boundary of the system of interest by defining the external actors who may interact with the system and the activities that each actor may participate in during system operation. The final step in development of Functional Architecture views is generation of a State Machine Diagram (Figure 42), which provides additional clarity regarding the range of behaviors possible for a given entity, as well as the differing modes of activities in different states. This allows for a more formal examination of the control system of the system of interest than is possible in the Sequence Diagram. For a discrete activity like Minehunting, the State Machine Diagram is certainly less impactful, however it does demonstrate to the users that there is a defined exit and entry from Minehunting during the activity (between Classification and Reacquisition) that is dependent on an external activity. It is impossible to complete the transition between all of the states of the Minehunting activity prior to completion of this external activity, which is not evident from the Activity, Sequence, or Use Case Diagrams. Note that while State Machine Diagrams can be used to define a behavior, this minimizes the utility of the figure by duplicating information and restricting freedom, and therefore it is recommended that State Machine Diagrams be used to describe the state dependent behaviors of physical components. This should facilitate development of an external model by defining capabilities and limitations on system behavior related to the current status of the system.

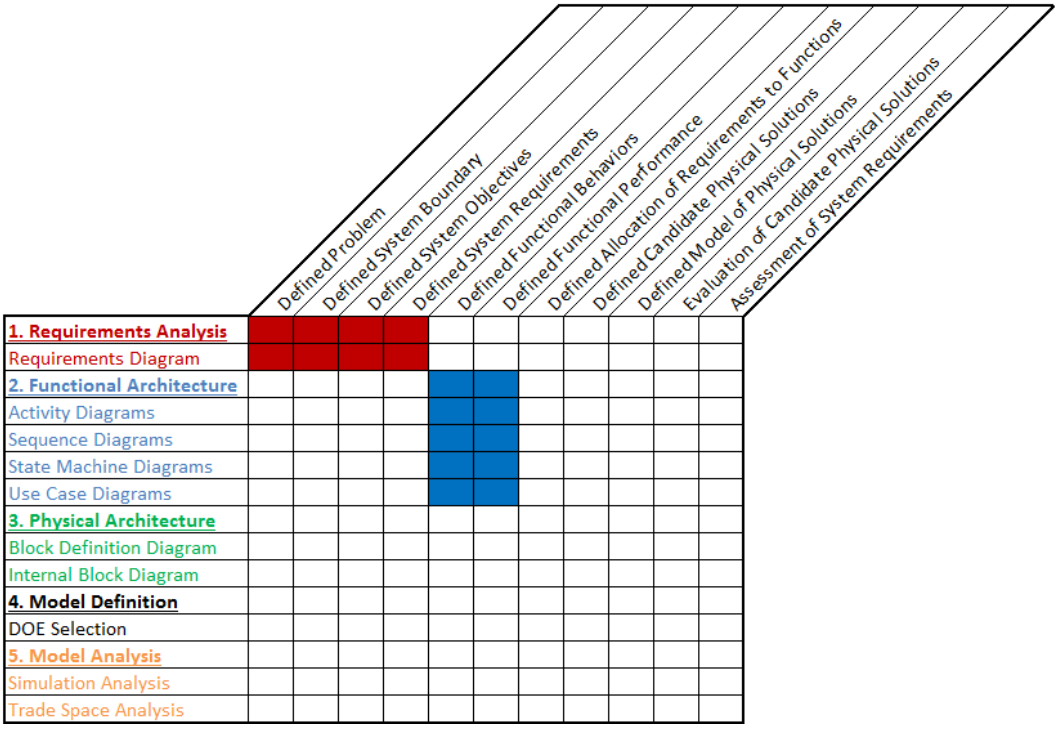
Figure 42 State Machine Diagram: Perform Mine Hunting Operations



Note that the coloring presented in Figure 42 is not typically presented in State Machine Diagram; however the author feels that the addition of colors aided understanding of the figure. Specifically, red coloring shows transitions that cause termination of minehunting operations, green coloring shows transitions from the first portion of the minehunting sequence (Detection and Classification) to the second portion of the minehunting sequence (Reacquisition and Identification), and yellow coloring shows transitions within each sub portion of the minehunting sequence. This movement within the functional architecture products toward describing the functions performed by different physical components (which at this point should still be kept as solution neutral as possible) suggests that a comprehensive description of the system must move to a more detailed description of those physical components. The completion of Activity, Sequence, Use Case, and State Machine Diagrams defines the system comprehensively from a functional perspective. The system architect can completely describe the sequencing of system activities, which can be used as a basis for model development. The system architect can also formally present limitations to system performance, whether they arise from some alteration to system environmental conditions (as identified in a State Machine Diagram), some issue with personnel availability (as identified in Use

Case Diagrams), some issue with system control (which, depending on implementation, may be represented in either a State Machine or Sequence Diagram), that must be represented in an external model. Table 3 presents an updated linkage of the systems engineering products supported by Step 2 (Functional Architecture) of the MBSE MEASA. Note that Table 3 suggests that only a single Functional Architecture SysML Diagram is required to support Defined Functional Behaviors and Defined Functional Performance, the intent of this section is to emphasize that each of the Functional Architecture SysML Diagrams provide a unique capability and each of the diagrams should be created in this MBSE MEASA Step.

Table 3 Functional Architecture Support of Linkage of MBSE MEASA Steps to Systems Engineering Products



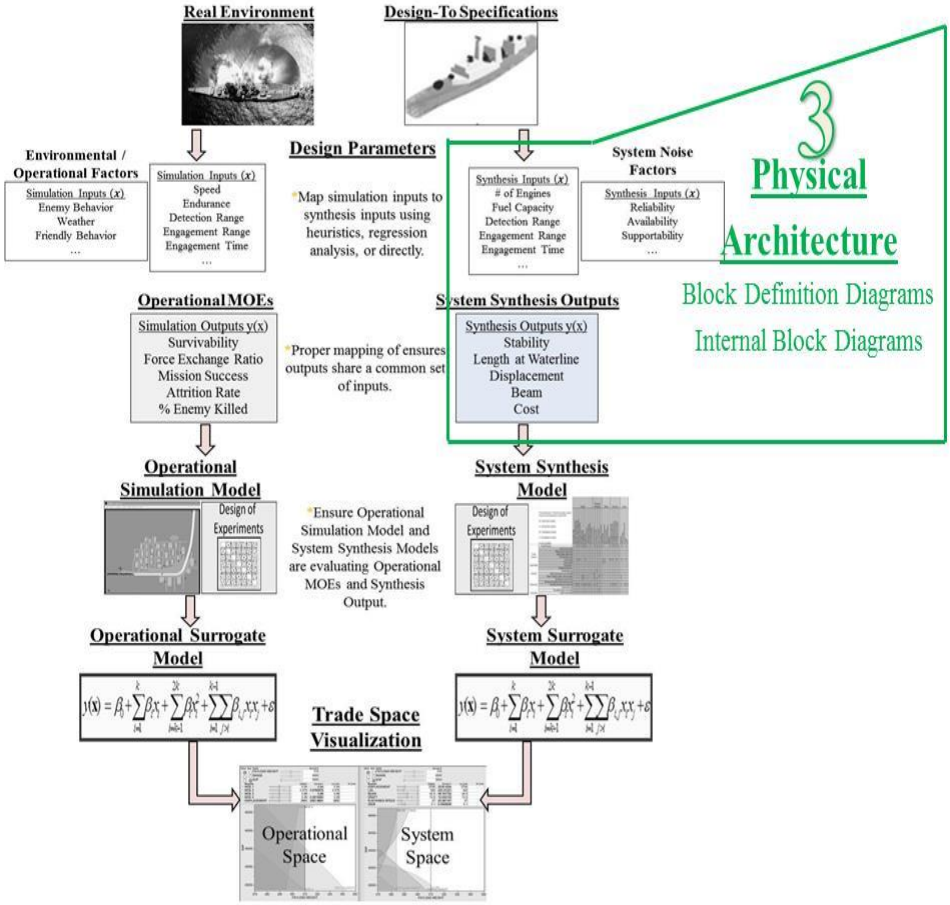
The Functional Architecture products describe the system in terms of functional behaviors and performance and ensure that external models and simulations accurately represent the activities that the system must perform, the sequence of those activities, the actors that should perform and impact those activities, and the transitions between those

activities. Because each of these SysML products are explicitly linked to the previously developed Requirement Diagram, any behaviors and activities represented in external operational simulations are directly linked to stakeholder input, and no extraneous behaviors are modeled and no fundamental system behaviors are ignored.

6. Physical Architecture Products

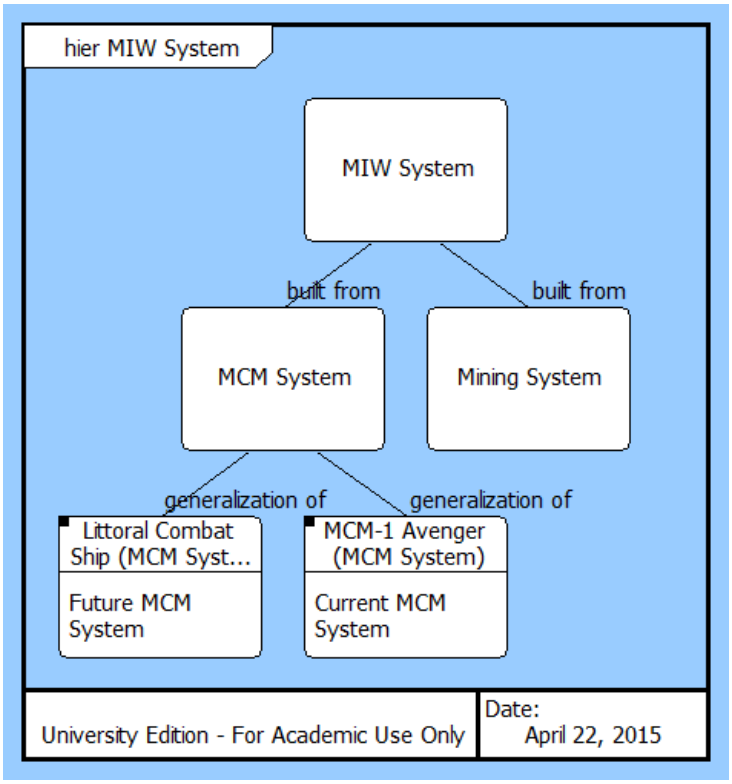
Completion of the functional architecture (which is now represented in Activity Diagrams, Sequence Diagrams, Use Case Diagrams, and State Machine Diagrams) triggers creation of Physical Architecture Products. These products are necessary to more completely describe the system and enable creation of external models. Figure 43 provides a description of the SysML products appropriate to support physical architecture development.

Figure 43 MBSE MEASA (Step 3)



Creation of a Block Definition Diagram is the first step in Physical Architecture development. Block Definition Diagrams decompose physical entities, which are only shown in a general sense in each of the Functional Architecture products, into more detailed components. One of the major advantages of Block Definition Diagrams is that they allow complete representations of the potential physical configurations of a system, even if components are mutually exclusive. In the case of the MIW System, the easiest illustration of such a relationship is the MCM-1 Avenger and the LCS, which are classified as "generalizations of" the MCM System component, indicating that they completely describe the MCM System of interest but cannot both exist in a given physical configuration. Conversely, the MIW System component is "built from" the MCM System component and the Mining System component, indicating that each exist for every configuration of a complete MIW System. Figure 44 provides a graphical representation of the Block Definition Diagram for the high level components of a MIW System.

Figure 44 Custom Block Definition Diagram: MIW System

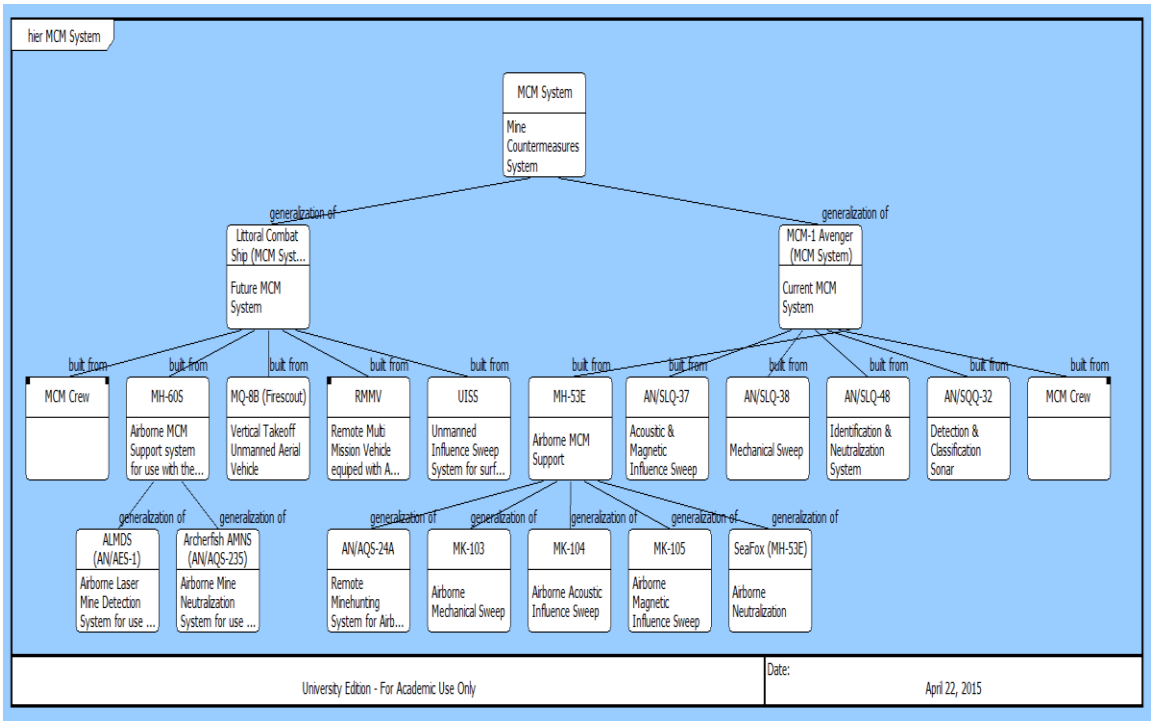


Note that this research presents a slightly altered version of the Block Definition Diagram in Figure 44. This is a result of a software limitation; the software selected for generation of SysML diagrams in this research (Vitech's CORE) does not allow for generation of traditional SysML Block Definition Diagrams. Rather, CORE requires users to create Structure Block Definition Diagrams (similar to a traditional physical hierarchy) and Classification Block Definition Diagrams (which represents the inheritance structure of each system component, similar to a UML Class Diagram). This convention segments "built from" and "built in" relationships (shown in the Structure Block Definition Diagrams) from "generalization of" and "generalizes" relationships (shown in the Classification Block Definition Diagram). Practically, this reduces the potential for incorrect relationship specification within the CORE software since it requires a user to truly understand the differences between the naming conventions. The power and richness of the CORE tool ensures that the relationships are represented properly in each diagram and also ensures that the linkages are traceable to the previously created SysML products. Within the software itself, separating Block Definition Diagrams into structural and classification perspectives is not a limitation, however for a user who wishes to present static representations of those diagrams (as in this research) it requires a user to create a custom diagram to fully capture both the structural and classification relationships between system components as a traditional SysML Block Definition Diagram.

This research develops custom SysML Block Definition Diagrams that combine the information that CORE typically presents as a Structure Block Definition Diagram and a Classification Block Definition Diagram. Figure 44 presented a simple example of a custom generated Block Definition Diagram and Figure 45 presents a more detailed example that decomposes the airborne components of both the MCM-1 Avenger MCM System and the Littoral Combat Ship MCM System (note that within the software these relationships were established within the Structure and Classification Block Definition Diagrams to check that each relationship was properly defined). Block Definition Diagrams allow the user to decompose each system component in as much detail as is necessary for the system of interest. In the case of the MIW System, the decomposition

continued through identification of all systems and subsystems, but did not decompose each subsystem into physical components. This decision is a result of the mature nature of the systems of interest, currently they have been completely designed (and many of the systems have been operated for thirty years). Within the context of the MBSE MEASA Block Definition Diagrams inform the physical components represented in external models and simulations (typically operational, physical, and cost models). Accordingly, the MBSE MEASA recommends that development of Block Definition Diagrams proceed to a sufficient level of detail to ensure that the physical components represented in external models and simulations can be checked for consistency with the physical components represented in Block Definition Diagrams.

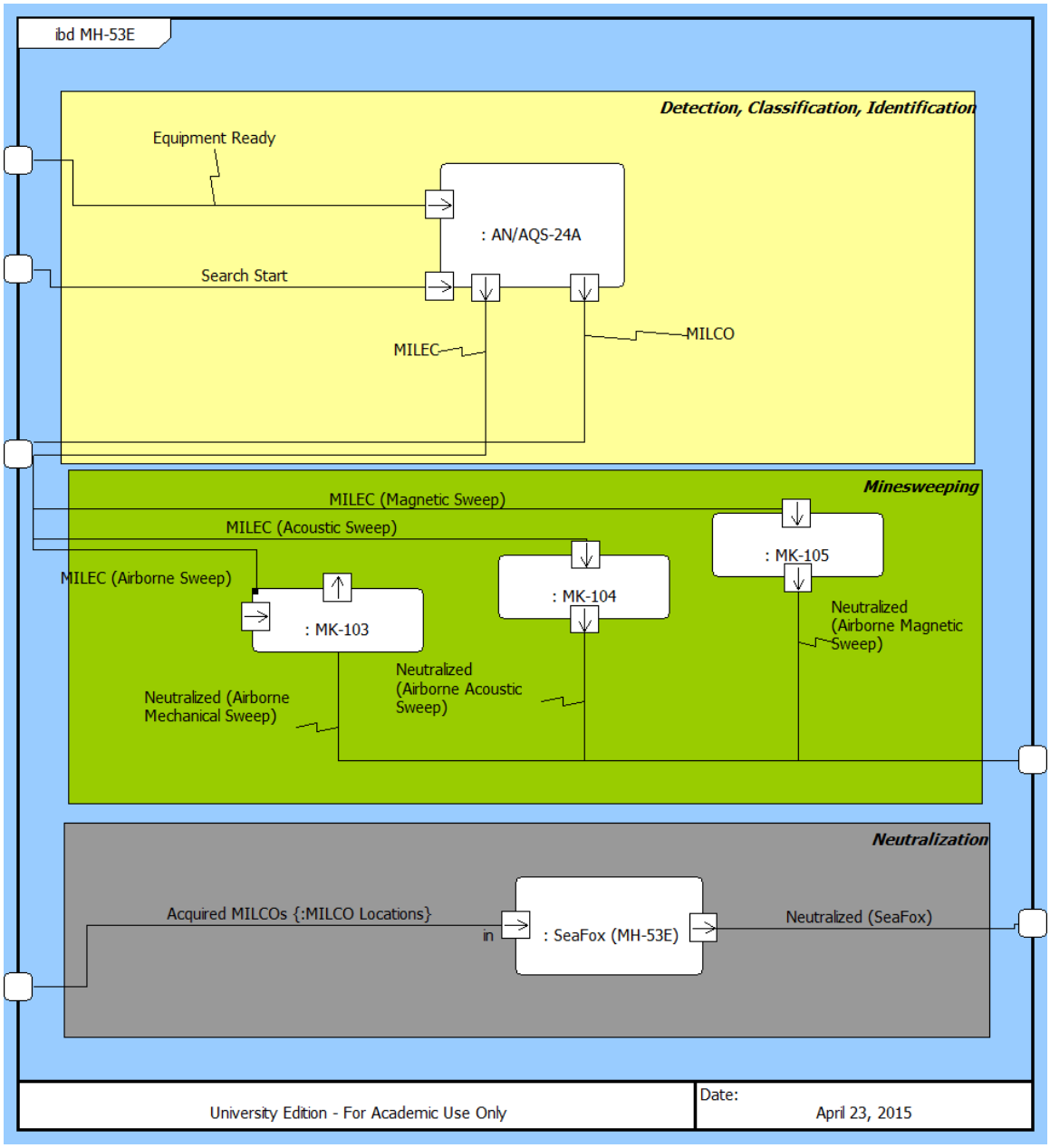
Figure 45 Custom Block Definition Diagram: MCM System



The Physical Architecture information captured in the Block Definition Diagrams defines the physical systems that exist in each potential system configuration. Internal Block Diagrams expand this functionality by establishing a connection between Block Definition Diagrams and Activity Diagrams by specifying how the elements shown in the

Block Definition Diagrams perform the activities shown in Activity Diagrams to achieve the intended functionality of the system. The major difference between the Internal Block Diagram and the Activity Diagram is the system perspective of the diagrams. As indicated by their grouping within the MBSE MEASA, Internal Block Diagrams present the system from a physical/structural perspective while Activity Diagrams present the system from a functional/behavioral perspective. Similarly, the simultaneous examination of Activity Diagrams defines the behaviors represented in an external model while Internal Block Diagrams defines the physical entities represented in an external model. Figure 46 shows an Internal Block Diagram for the MH-53E (which conducts airborne MCM in support of the MCM-1 Avenger MCM System). Note that the MH-53E is capable of performing the full sequence of mine detection through mine neutralization (although it must use a different subsystem when performing mine detection through identification and when performing mine neutralization).

Figure 46 Internal Block Diagram: MH-53E



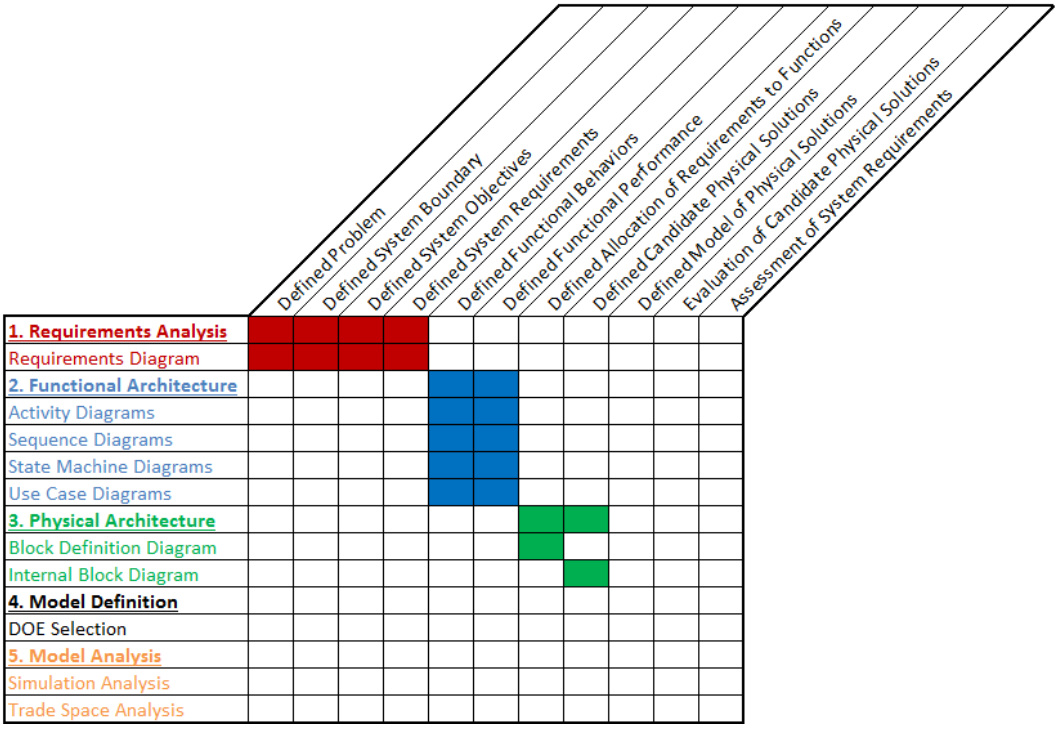
Note that Figure 46 adds organizational blocks (a yellow block for Detection, Classification, Identification; a green block for Minesweeping; and a grey block for Neutralization), that are not necessary components of Internal Block Diagrams. The author added the organizational blocks to aid visualization and the blocks do not need to be added in situations where the Internal Block Diagram is simple enough that they add

no value (as a general guidance, this will most likely be for system components that only perform one or two activities represented in Activity Diagrams). Figure 46 highlights the utility of the Internal Block Diagram, specifically the definition of the physical components that make up larger components (not dissimilar for the Block Definition Diagram) and also the establishment of the interfaces between components and the links between each component and components external to the system of interest. For example, Figure 46 shows that the AN/AQS-24A sonar links to an external system through the creation of MILECs and MILCOs. That same external system (represented on the left side of Figure 46) links to the MK-103, MK-104, and MK-105 minesweeping systems. While it is impossible to show within an Internal Block Diagram for a single system, an examination of the set of links between blocks within the modeling software establishes that the AN/AQS-24A, MK-103, MK-104, and MK-105 are all linked the Post Mission Analysis component, which is processing the list of MILECs and MILCOs and determining which component is appropriate to process each potential mine. Internal Block Diagrams can be produced for any level of detail necessary for a given system. As with Block Definition Diagrams, it is recommended that Internal Block Diagrams be produced at a sufficient level of detail to define the physical components and interfaces that should be represented in an external model. Given that the purpose of this study is to examine the performance capabilities of the MCM-1 Avenger MCM System and the Littoral Combat Ship MCM System, these Internal Block Diagrams represent the interfaces between components at the subsystem level. If this study was comparing alternative capabilities for mine sonar systems, Internal Block Diagrams could decompose the AN/AQS-24A into its components (propulsion system, detection system, etc.) and establish the interfaces between those components. It is left to the individual user to ensure that all physical architecture products (Block Definition Diagrams and Internal Block Diagrams) contain enough detail to accurately construct external operational, synthesis, and cost models.

Table 4 presents an updated linkage of the systems engineering products supported after Step 3 (Physical Architecture) of the MBSE MEASA. Note that the use of Block Definition Diagrams supports definition of a complete set of physical solutions

while the Internal Block Diagrams are focused on ensuring allocation of physical components to system functions (as represented in the previously developed functional architecture products, most notably the Activity Diagrams).

Table 4 Physical Architecture Support of Linkage of MBSE MEASA Steps to Systems Engineering Products



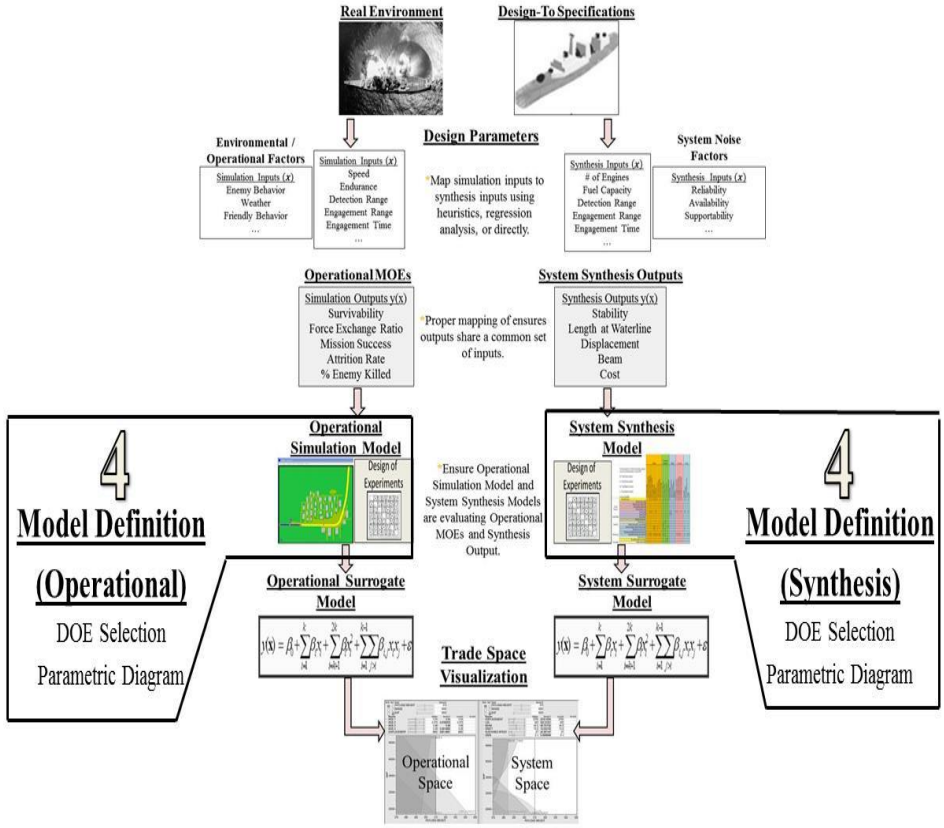
7. Modeling and Simulation Definition

Completion of physical architecture development (creation of Block Definition and Internal Block Diagrams) completely describes a system. The Requirement Diagram completely describes what a system must do to satisfy stakeholders. The Activity, Sequence, Use Case, and State Machine Diagrams completely describe how a system satisfies the requirements identified in the Requirement Diagram. The Block Definition and Internal Block Diagrams completely specify what physical system components satisfy the functions specified by the Activity, Sequence, Use Case, and State Machine Diagrams. While these products enable a complete description of a system from multiple

perspectives, they do not enable analysis of the performance of that system. Proper analysis requires definition and analysis of external system models.

At this point the true utility of the MBSE MEASA (beyond generation of SysML products or systems engineering process products) starts to become apparent. Rather than conduct system analysis by attempting to link physical/functional architectures through Parametric Diagrams (as is advocated by the MBSE methodologies presented in Chapter II), the MBSE MEASA separates external Model Definition as Phase 4 of the process (Figure 47). Note that the MBSE methodologies presented earlier consider the creation of a comprehensive set of SysML products (which have been developed by the conclusion of the first three steps in the MBSE MEASA), along with evaluation for consistency through the use of Parametric Diagrams, the conclusion of the system development process. The MBSE MEASA utilizes the combined functional and physical architecture products as a basis for the development of external models and simulations. In particular, the Activity, Sequence, Use Case, and State Machine Diagrams specify the components, behaviors, and processes represented in external operational models. The Block Definition and Internal Block Diagrams specify the set of components, component interfaces, and constraints represented in external system synthesis models and system cost models.

Figure 47 MBSE MEASA (Step 4)



The formalization of an analysis procedure external to the SysML modeling process prevents the types of oversimplifications of system performance that occur if sufficiently detailed modeling of system performance is not conducted. As mentioned previously, SysML Parametric Diagrams typically support modeling and simulation in the leading MBSE methodologies. While Parametric Diagrams can be useful to ensure consistency between functional and physical architecture products, their limitations must be acknowledged.

To conduct rapid decision making regarding preferred system configurations Parametric Diagrams often make simplifying assumptions about each potential system configuration. Parametric Diagrams only examine the ability of a given system configuration (characterized by defined system component performance) to successfully complete the activities specified in associated SysML diagrams. While this often allows

for rapid evaluation of system configurations, it effectually over-constrains the problem and makes it impossible to truly assess the operational performance of a system. Various MBSE methodologies acknowledge this limitation, but the fundamental approach that those methodologies advocate regarding system performance modeling is still problematic. A simple example from Friedenthal, Moore, and Steiner (2009) espouses the usage of an external model to analyze the engine type (V6 versus V4) required to satisfy a vehicle system requirement for acceleration. The acceleration requirement is subject to constraints on Gravitational Force, Drag Force, Power Train Force, Total Force, Engine Torque, Transmission Torque, Differential Torque, and Wheel Force, which are all represented as constraints (implemented as specified constant values) within a parametric diagram. The diagram is subsequently executed and examined to determine what engine type is most appropriate. The example concludes that "the analysis results showed that the V6 configuration is needed to satisfy the vehicle acceleration requirement." While this is useful to support engineering level analysis on specific system configurations, the simplicity of that statement highlights the shortcomings associated with over-constraining a problem through the use of detailed SysML Parametric Diagrams. Perhaps a V4 configuration would be capable of satisfying the performance requirement if the body type, wheel type, chassis type, etc., were changed. By specifying values for each of these (potentially) impactful variables earlier in the system design process the number of system configuration alternatives that may be examined is limited. Limiting the amount of raw data generated subsequently limits the range of potential conclusions. Accordingly, the MBSE MEASA does not recommend that system analysis within an MBSE methodology rely solely on the creation and analysis of SysML Parametric Diagrams. Rather, the MBSE MEASA recommends the creation and analysis of external simulations based on the set of SysML products developed previously in the methodology.

Many types of external simulation, ranging from process based to agent based simulations may be appropriate to support analysis within the context of the MBSE MEASA. It is the responsibility of the user to select an appropriate simulation, however in practice the selection of an "appropriate" simulation may be quite difficult. Practically

this decision will be highly dependent on the expertise of the particular user, and the selection of the type of model may be based almost exclusively on this expertise. However, it is useful to provide some references within this dissertation that provide in depth discussion of the strengths and weaknesses associated with different types of simulation models. Perhaps the most widely read text that discusses the procedures and characteristics of process based and agent based models is Law (2014), which is essential reading for any simulation developer. Law (2014) certainly focuses more detail on the development and analysis of discrete event simulations, for more concise guidance on agent based simulation Macal and North (2005) provide an introduction to the principles and expected applications of agent based models. As mentioned, this research does not provide a complete description or recommendation of a particular simulation modeling paradigm, rather it emphasizes that an appropriate simulation modeling approach must be selected and tailored for each study and provides a few references to guide the selection of an "appropriate" simulation model, while recognizing that the choice is often reduced to the familiarity and expertise of a particular user.

After an appropriate simulation has been chosen to support the MBSE MEASA, proper testing procedures for those models and simulations must be defined. Substantial work has been done in the field of experimental design that must be reviewed by any user implementing the MBSE MEASA.

8. Experimental Design Recommendations

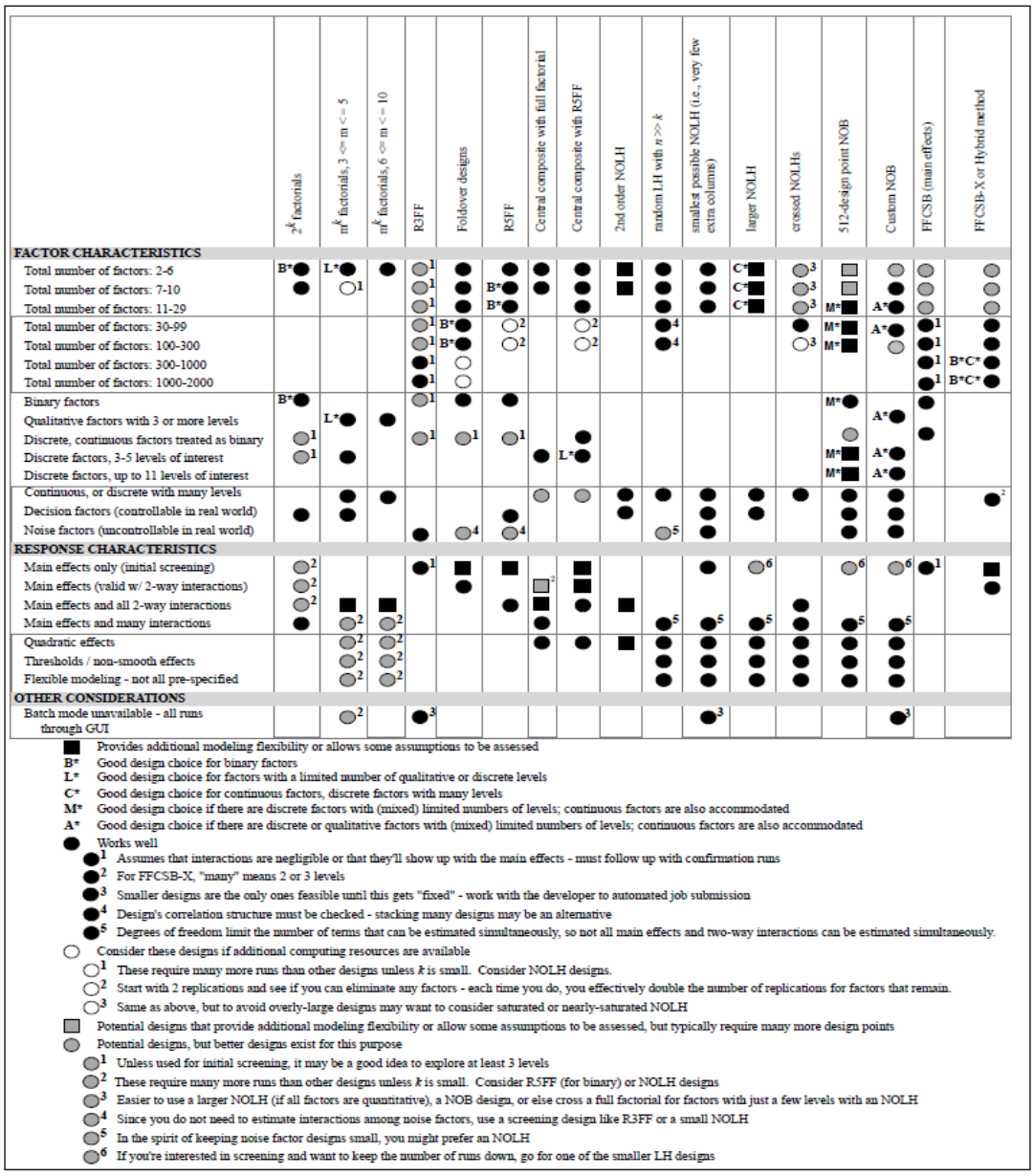
As mentioned in Chapter II, existing MBSE methodologies, as well as recent research in MBSE, fail to emphasize the importance of proper experimental design selection in the development of external models and simulations to support MBSE focused system development. An exception is MacCalman et al. (2015), which provides a case study analysis of a U.S. Army infantry squad that demonstrates the value of proper experimental design specification in a MBSE approach. However, a discussion of experimental design is necessary in this dissertation to establish guidelines for application of the MBSE MEASA.

Experimental design selection is vital to ensure that alternative system configurations are examined properly. Simply establishing a baseline system configuration and conducting testing and evaluation through isolated excursions is inappropriate (see Sanchez and Wan 2012). As discussed in Giammarco and Auguston (2013), it is vitally important to consider system component interactions as well as the set of possible interactions between a system and its environment. The formalization of testing procedures that ensures that system configurations and potential interactions are considered falls under the category of experimental design. While a detailed review of experimental design is not the focus of this dissertation, Appendix B provides a brief introduction to experimental design for the unfamiliar reader and highlights the consequences associated with establishing a baseline system and testing through isolated excursions. More details on the fundamentals of experimental design are provided in Montgomery (2012) and Myers, Montgomery, and Anderson-Cook (2009). Detailed discussion of experimental design for computer experiments can be found in Santner, Williams, and Notz (2003), and guidelines for the implementation and analysis of simulation models (to include a brief review of experimental design, as well as further detail regarding the differences between agent based and discrete event models) can be found in Law (2014).

Sanchez and Wan (2012) present a focused discussion of the guidelines for proper experimental design selection for simulation experiments. That research addressed the challenges associated with different factor types, specifically quantitative vs qualitative factors, discrete vs continuous factors, and controllable vs uncontrollable factors. That research demonstrates that space filling experimental designs offer tremendous advantages over traditional factorial experimental designs for computer experiments, specifically in terms of the number of variables that may be considered (this issue is discussed in Appendix B) and the tremendous flexibility that space filling designs offer in terms of model fitting (while traditional experimental designs typically restrict model fitting to linear or quadratic models, space filling designs impose almost no restrictions on model fitting). In the context of this research, the experimental design comparison chart found in Sanchez and Wan (2012) guides the type of experimental design

appropriate for a given number of input factors, the characteristics of those factors, and the desired type of model fit (Figure 48).

Figure 48 Experimental Design Comparison Chart



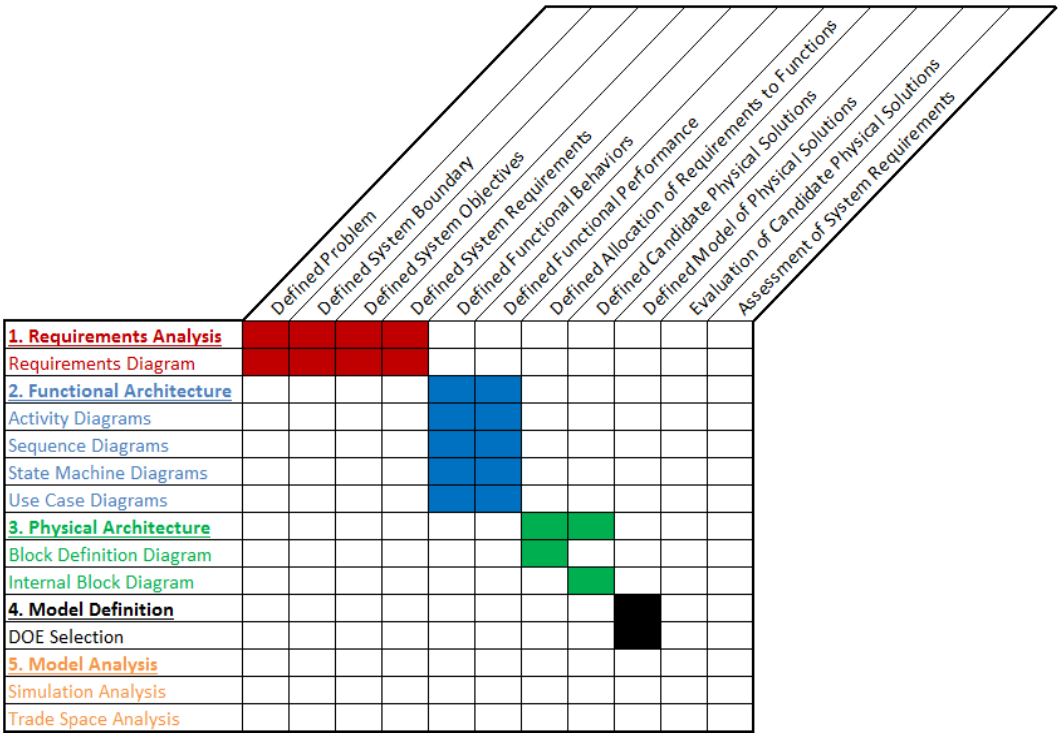
Source: Sanchez, Susan M., and Hong Wan. 2012. "Work Smarter, Not Harder: A Tutorial on Designing and Constructing Simulation Experiments." S*imulation Conference (WSC), Proceedings of the 2012 Winter Simulation Conference*, 1–15.

Figure 49 may be used as a starting point for selection of experimental design for almost any scenario where testing is conducted in a simulation model. Examination of the chart key presented in the lower portion of Figure 48 suggests that designs represented with a black square should work exceptionally well (that is, they provide maximum modeling flexibility) for examination of large scale, complex systems, which are comprised of a large number of components and have the potential to exhibit higher order interactions. This aligns nicely with the definition of the systems of interest to this research presented earlier. Furthermore, because the systems of interest to the methodology typically contain at least 100 factors it is potentially dangerous to assume that their behavior can be characterized through simple linear or quadratic models. Accordingly, utilization of designs that can examine at least 100 factors that provide maximum modeling flexibility is desirable.

Examination of Figure 48 suggests that 512 design point NO/B designs are appropriate for these scenarios (note that NO/B stands for Nearly Orthogonal/Balanced). Those designs are discussed in detail in Vieira et al. (2011) and Vieira et al. (2013), which presents a mixed integer programming approach for the generation of experimental designs for discrete and continuous factors. The ability of these 512 design point NO/B designs to consider both discrete and continuous factors is vitally important when considered large scale, complex systems. Because these types of systems may include components that can only take defined, discrete values (ex: an on/off factor can only take two values, a high/medium/low factor can only take three values, etc.) it is valuable to use experimental designs created specifically for these types of factors. Vieira et al. (2013) details the issues associated with choosing an experimental design created only for continuous factors and rounding the values of each design point, specifically it reduces the orthogonality of the designs. Additional information on the latest design and analysis techniques for large-scale simulation experiments, as well as over 150 examples of their application to problems in defense and homeland security can be found at the Simulation Experiments & Efficient Designs (SEED) Center for Data Farming's web pages at harvest.nps.edu.

Table 5 presents an updated linkage of the systems engineering products supported after Step 4 (Model Definition) of the MBSE MEASA. Note that the sole focus of Step 4 is creating external models and simulations of potential physical solutions (meaning that the objects represented in both the external operational and system synthesis models and simulations are defined by the physical solutions described by the Physical Architecture SysML Diagrams developed in Step 3 of the MBSE MEASA). This facilitates analysis of system performance beyond the capabilities of SysML Parametric Diagrams and, after appropriate analysis of results is conducted, establishes a quantitative linkage between operational MOEs and system design parameters as well as between system design characteristics and system design parameters. Previously developed SysML products define the activities and entities included in these models and the experimental design techniques prescribed in this section guide the testing of these models and simulations.

Table 5 Model Definition Support of Linkage of MBSE MEASA Steps to Systems Engineering Products



9. Model Analysis

While experimental design is vital to definition and analysis of all models and simulations, the presentation of those analysis results is also extraordinarily important in the context of modeling and simulation. The final step of the MBSE MEASA (Figure 49) is presentation and analysis of the results of simulation models. The analysis assesses how well various Physical Architecture combinations (from Step 3) satisfy the Functional Architecture (Step 2) defined system performance. Sitterle et al. (2015) advocate this approach, demonstrating an interactive tool that enables analysts "to quickly and accurately assess and compare alternatives" supports consistent, analytical, traceable decision making. Creation of a dynamic dashboard, as presented earlier in this research, is a demonstrated method that supports the MBSE MEASA and facilitates traceable decision making. Such an approach rapid visualizes of system level trade-offs and facilitates discussion of potentially conflicting system requirements based on both operational and system level models and simulations. As mentioned previously, MacCalman et al. (2015) present modeling results for a U.S. Army simulation in a dynamic fashion. That work, details instructions regarding the utility of dynamic decision making displays, also guides the definition and creation of these displays. The objective of this research is not to provide a formal definition of analysis procedures or instructions for creation of dynamic decision making displays. Accordingly this research does not present explicit guidelines, although the process prescribed in MacCalman et al. (2015) is a valuable starting point.

Figure 49 MBSE MEASA (Step 5)

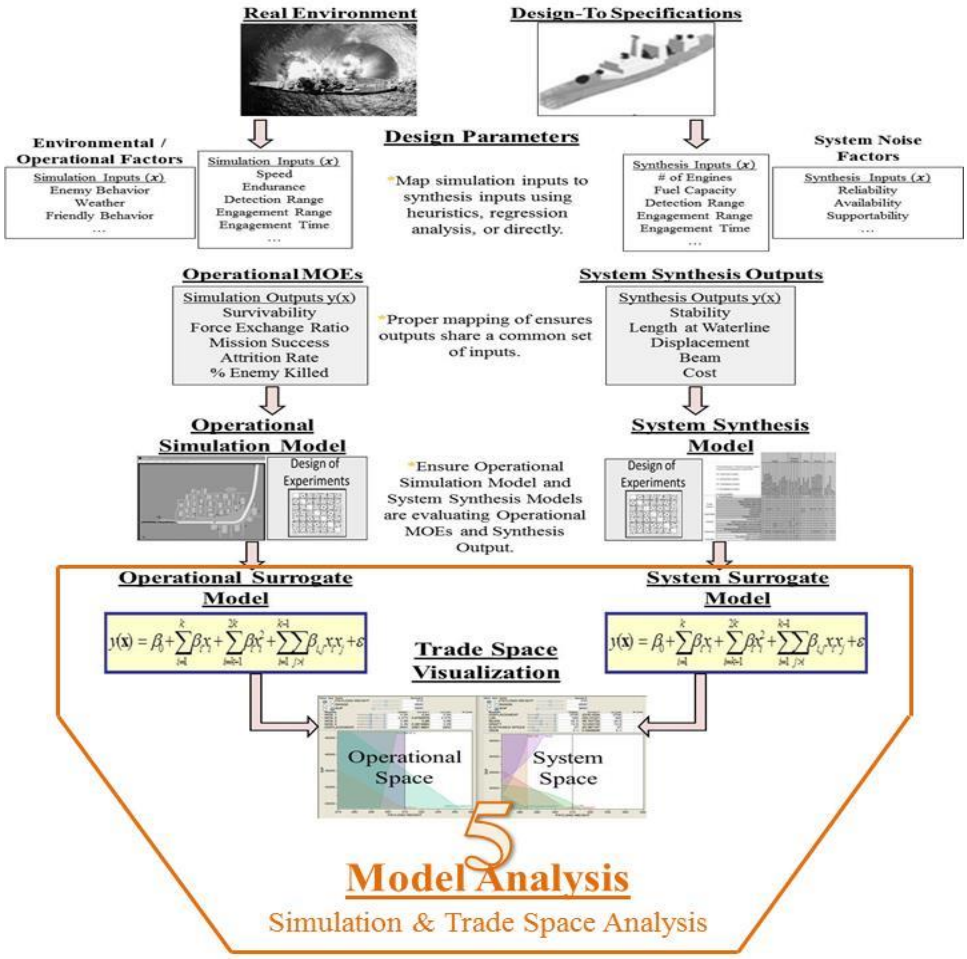
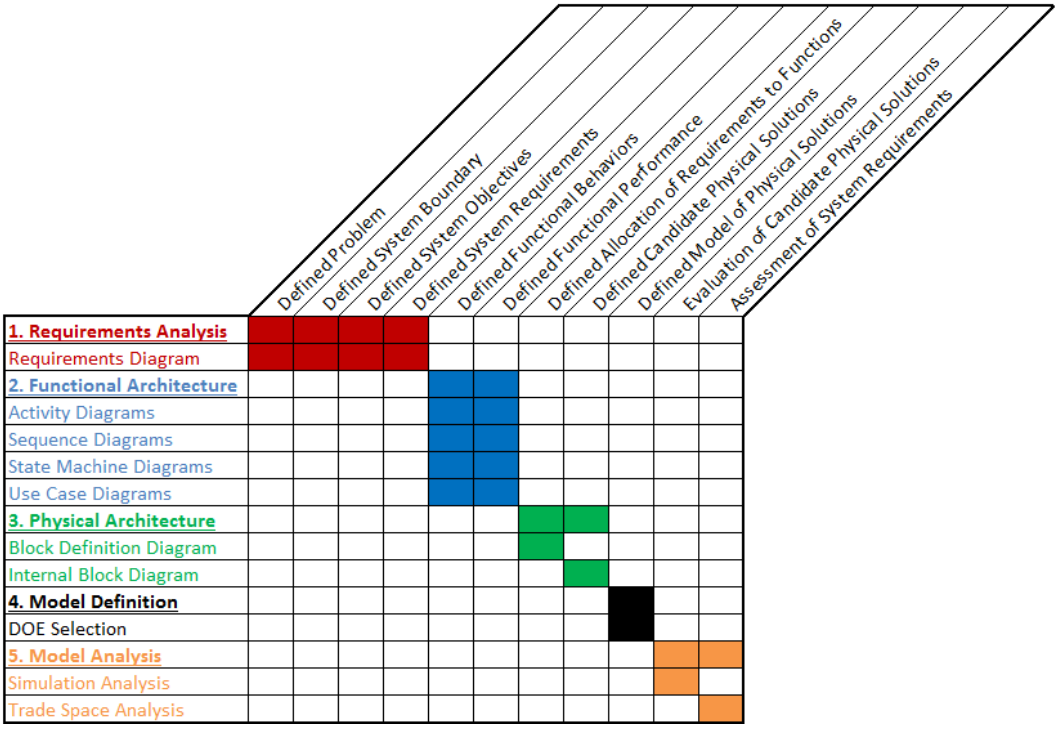


Table 6 presents an updated linkage of the systems engineering products supported after Step 5 (Model Analysis) of the MBSE MEASA is complete.

Table 6 Model Analysis and Analysis Iteration Support of Linkage of MBSE MEASA Steps to Systems Engineering Products



C. MBSE MEASA ITERATION

As mentioned in Chapter II, one of the major contributions of the MBSE MEASA is an explicit focus on the iteration of the methodology to demonstrate not only how system architecture supports system analysis, but also how system analysis results can be incorporated into existing system architecture products to refine subsequent system analysis. While there are numerous approaches to ensuring consistency within system architecture products, within system models, and within system analysis results, the MBSE MEASA presents a framework and guidelines for ensuring consistency across these domains. To ensure consistency with the five steps of the MBSE MEASA, the iteration of the methodology will focus on appropriate integration of system analysis results into SysML products.

Recall that one of the primary emphases of the MBSE MEASA is that system architecture and analysis must incorporate and examine system design variables, system

operational variables, and system environmental variables. Equally importantly, the MBSE MEASA emphasized that potential interactions between variables (either within or between categories) must be recognized. Generally, this allows for nine potentially impactful variable relationships that can be identified during system analysis that must be represented in future iterations of the system architecture (assuming that the analyst and stakeholders are interested in identifying these relationships explicitly, rather than using a robust design approach to develop systems that are inherently robust to variation in environmental and other uncontrollable variables). Table 7 presents a visual representation of these nine potentially impactful cases, grouped according to the variable type of interest and the analysis results. For brevity the cases are coded, Cases 1a, 1b, and 1c correspond to analysis results indicating that a single design, operational, or environmental variable impacts system performance. Cases 2a, 2b, and 2c correspond to analysis results indicating that there are interactions between design variables, interactions between operational variables, and interactions between environmental variables that impact system performance. Case 3a corresponds to impactful interactions between design and operational variables, Case 3b corresponds to impactful interactions between operational and environmental variables, and Case 3c corresponds to impactful interactions between environmental and design variables. Note that the numbering is introduced to aid organization and does not imply that Case 1 relationships are inherently more important that Case 2 or Case 3 relationships.

Table 7 Listing of Analysis Result-Variable Type Cases Requiring MBSE MEASA Iteration

Analysis Result	Variable Type: Design	Variable Type: Operational	Variable Type: Environmental
Main Effect	*Case 1a:* Design	*Case 1b:* Operational	*Case 1c:* Environmental
In Category Interaction	*Case 2a:* Design - Design	*Case 2b:* Operational - Operational	*Case 2c:* Environmental - Environmental
Between Categ Interaction	*Case 3a:* Design - Operational	*Case 3b:* Operational - Environmental	*Case 3c:* Environmental - Design

The classification of a variable as design, operational, or environmental is often intuitive, but general definitions are necessary to provide clarity regarding the definition of variables in the context of the MBSE MEASA. Recall that the MBSE MEASA assumes that the system of interest is being examined within a simulation model, where every variable is controllable (even variables such as the impact of the environment, which are not controllable in reality, are controlled and specified in the simulation model). The controllable nature of every variable within the simulation makes classification of variables important. Sanchez (2000) and Santner, Williams and Notz (2007) present variable definitions for simulation models, focusing primarily on whether or not a variable that is controllable in the simulation model is practically controllable in the real world environment. Accordingly, the MBSE MEASA follows a similar grouping convention to the definitions presented in Sanchez (2000). Specifically, the MBSE MEASA classifies design and operational variables as decision factors (other literature classifies these types of variables as control, engineering, or manufacturing variables), defined by Sanchez (2000) as factors "which are controllable in the real world setting modeled by the simulation" (70). The MBSE MEASA further segments decision factors

into design and operational variables, where a design variable refers to a design parameter within the control of the systems engineer that describes the configuration of the system, and an operational variable is within the control of the systems engineer and describes the operation of the system. The MBSE MEASA relies on the Sanchez (2000, 70) definition of noise factors as "not easily controllable or controllable only at great expense" to develop the characteristics of environmental variables. In the context of the MBSE MEASA, an environmental variable is outside of control of the systems engineer and potentially impacts the operation of the system.

1. Iteration of MBSE MEASA for Significant Main Effects

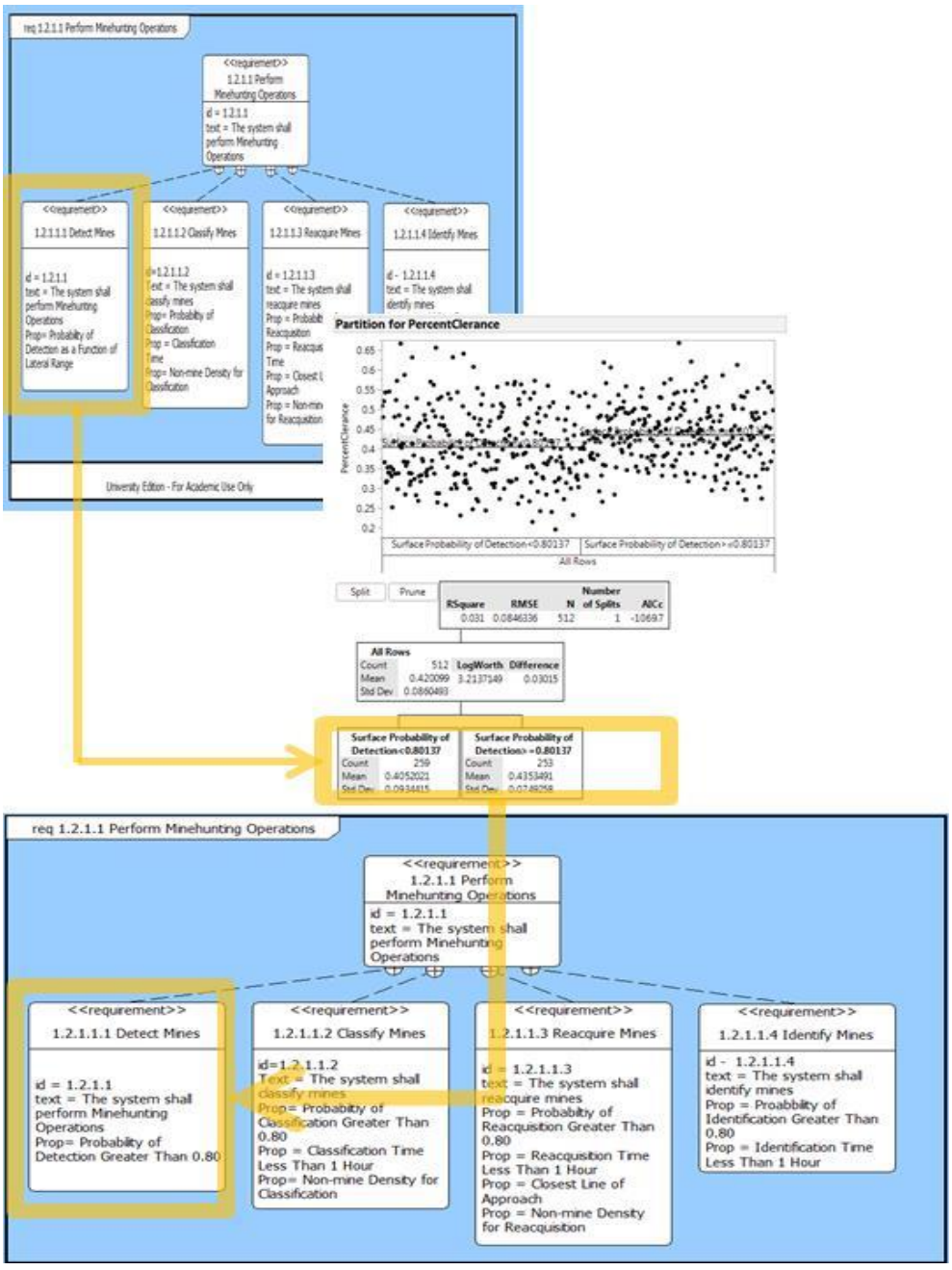
This section presents guidelines and illustrative examples demonstrating how impactful main effects identified in Step 5 of the MBSE MEASA can be introduced in future iterations of the MBSE MEASA. This section will provide three illustrate examples, one describing appropriate integration of impactful design variables (Case 1a), one describing appropriate integration of impactful operational variables (Case 1b), and one describing appropriate integration of impactful environmental variables (Case 1c).

a. Iteration of MBSE MEASA for Impactful Design Variables

Case 1a corresponds to situations when simulation model analysis suggests that a design variable has an impact on system performance. Because the MBSE MEASA advocates the creation and definition of a comprehensive Requirement Diagram as Step 1 of the process, the integration of this result into a future iteration of the MBSE MEASA is straightforward. This demonstration continues the example of the Active, Defensive MCM system and provides an example of the procedure that can be used to integrate such a result into a Requirement Diagram. This example assumes that analysis indicates that the Probability of Detection has been identified through analysis as impactful and that further analysis suggests that the Probability of Detection must be at least 0.80 for the system to achieve acceptable performance. Figure 50 provides a visual representation of how such a finding can be integrated into a SysML Requirement Diagram. Note that the analysis snapshot is purely notional, as stated, this example demonstrates iteration

when the effect of a given design variable is identified as potential impactful (in this case, the Probability of Detection).

Figure 50 Integration of Impactful Design Variable in Subsequent MBSE MEASA Iteration



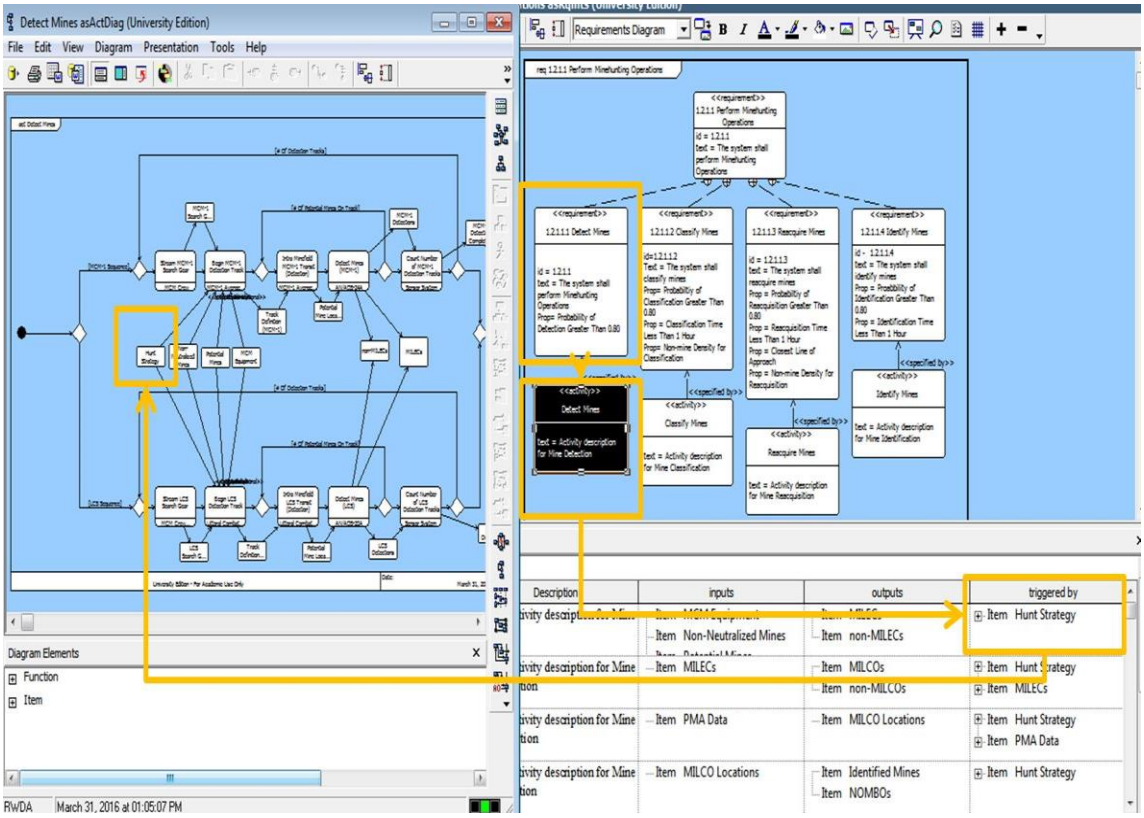
Note that Figure 50 presents a visual representation of the Requirement Diagram presented in Figure 35, a notional analysis result that suggests that the Probability of Detection should be set at a value greater than 0.80, and subsequently modifies the Requirement Diagram to expand the property for "Detect Mines" to specify that it should be at least 0.80.

b. Iteration of MBSE MEASA for Impactful Operational Variables

Case 1b corresponds to situations where simulation models analysis suggests that an operational variable has an impact on system performance. Integration of such a result may require additional alterations to the previously developed SysML products beyond editing of properties in SysML Requirement Diagrams. Note that there are scenarios where the integration of operational variables may mirror the example presented in the previous section on design variables (for example, the percentage of a minefield that is searched by one asset versus a second asset may be fixed in a SysML Requirement Diagram following the same procedure as used for the Probability of Detection shown in the design variable section). However, detailed integration of alterations for impactful operational variables most likely requires simultaneous consideration of both the Requirement Diagram and the Activity Diagram, which requires additional consideration of the relationships specified for each system requirement. As mentioned in Chapter II, Requirement Diagrams can represent c*ontainment*, *derive*, or *copy* relationships to expand requirements to requirements relationships as well as s*atisfy*, *verify*, *refine*, or *trace* relationships to relate requirements to system elements or activities. In particular, the *satisfy* relationship is particularly useful to coherently integrate impactful operational variables into future iterations of the MBSE MEASA. The satisfy relationship allows a user to specify that a requirement is satisfied by a model element other than another requirement. This allows a user to directly link a requirement (such as Detect Mines) to an activity (such as Detect Mines). The "Detect Mines" activity can then be expanded based on information contained in the "Detect Mines" requirement. Figure 51 presents an example of the implementation of a Requirement Diagram that has been expanded using a satisfy relationship (note that within the modeling software selected the *satisfy* relationship has been re-termed *specify*. The properties associated with the *specify*

relationship are exactly the same as the SysML *satisfy* relationship, future users may wish to select an alternative modeling program that is completely SysML compliant, users of CORE should be aware of this slight deviation from SysML convention. Note that this example (and the examples for each subsequent case) assumes a similar analysis procedure to the one highlighted for Case 1a has been conducted, but the analysis results will not be presented for each case.

Figure 51 Integration of Impactful Operational Variable in Subsequent MBSE MEASA Iteration (Requirement Diagram Satisfied by Activity Diagram Details)



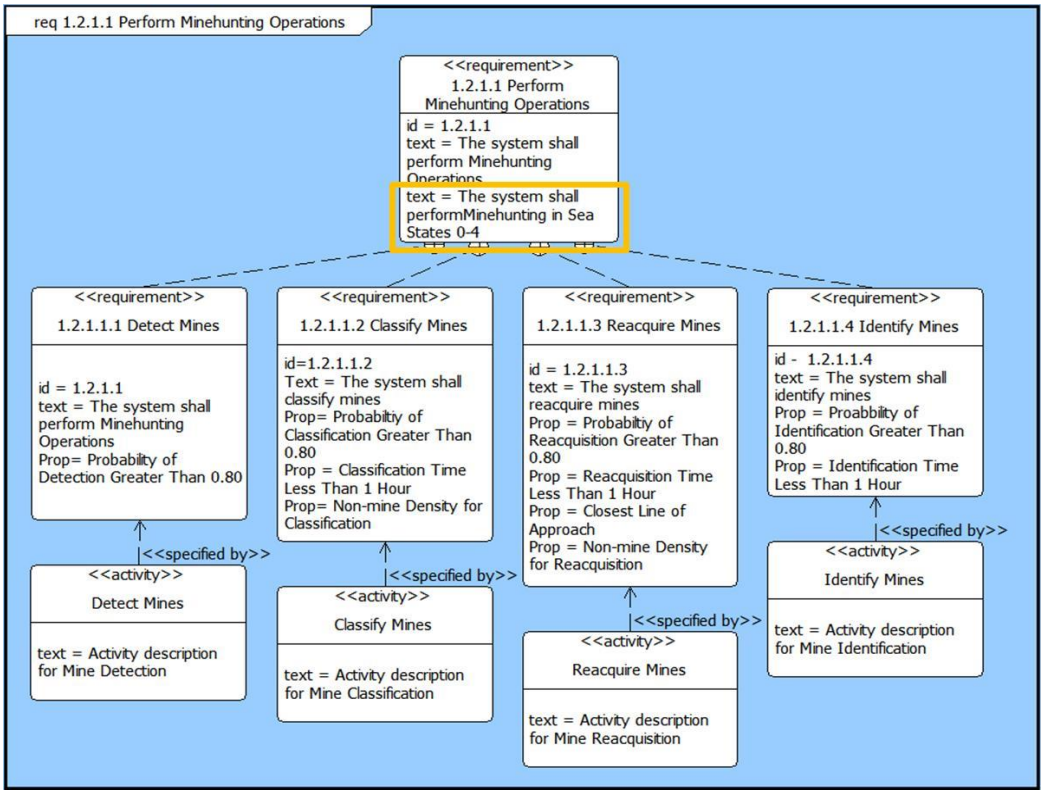
Each of the requirements shown is associated with an activity in the same manner as the "Detect Mines" requirement and the "Detect Mines" activity. The "Detect Mines" activity is expanded on the bottom right of Figure 51 to show the description, inputs, outputs, and triggers for the activity. Particularly important is the "Hunt Strategy" trigger, which specifies whether the activity utilizes the MCM-1 Sequence or the LCS Sequence

(an operational variable) as well as the number of Detection Tracks utilized in the activity (another operational variable). This explicit linkage between requirements and activities ensures that any operational variable findings identified in previous versions of the MBSE MEASA (such as a preference between the MCM-1 or the LCS or a preferred number of Detection Tracks) can be integrated completely and consistently in future iterations of the MBSE MEASA.

c. Iteration of MBSE MEASA for Impactful Environmental Variables

Case 1c corresponds to situations when analysis indicates that an environmental variable has an impact on system performance. Environmental variables are outside the control of the systems engineer, and therefore the user must take a more holistic view of the system. Recall that Step 1 of the MBSE MEASA advocated creation of context level system architecture products. This not only aided conceptual understanding of the system of interest but also explicitly defined the inputs and outputs to the system as well as the interactions between the system and the external environment. This facilitated development of high level SysML Requirement Diagrams, positioning the system requirements in terms of the broader operating concept. Because environmental variables have broad applicability to the system of interest, it is easiest to incorporate them into future iterations of the MBSE MEASA via higher level Requirement Diagrams (that is, specify that an environmental condition exists within a higher level requirement, thereby ensuring that it applies to each possible application of the system of interest). Figure 52 provides an example of the integration of an impactful environmental (in this theoretical example analysis has indicated that the system must conduct minehunting in Sea States 0–4) variable for Detect Mines using a higher level requirement (Perform Minehunting Operations).

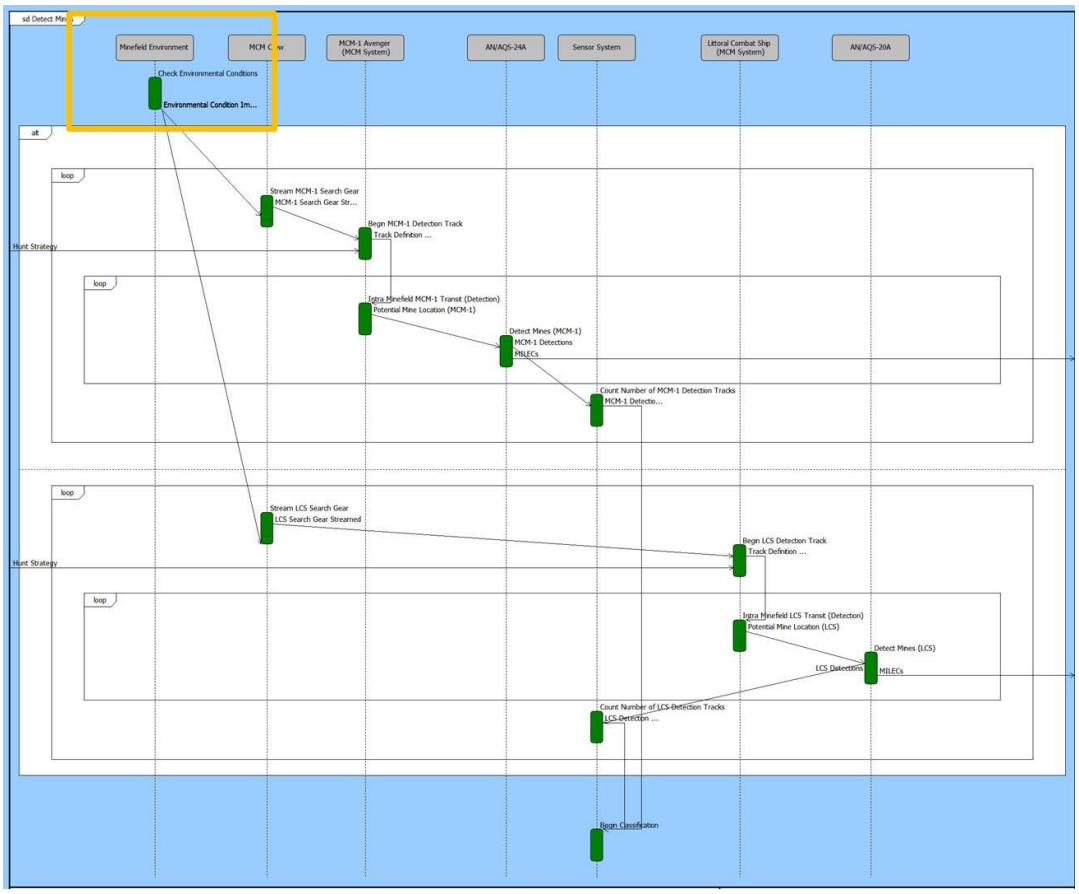
Figure 52 Integration of Impactful Environmental Variable in Subsequent MBSE MEASA Iteration (Inclusion of Environmental Condition in Higher Level Requirement)



After the Requirement Diagram has been expanded to include explicit reference to the importance of the environment on the system, it can be explicitly added to the SysML functional architecture products to ensure that it is properly represented in subsequent external models. While it is possible to environmental considerations to SysML Activity Diagrams (through a series of if-then decisions) or SysML Use Case Diagrams (while not intuitive, the environment could be represented as an external actor and its relationship with the system could be explicitly defined), the most thorough representation of the relationship between the environment and the system of interest is achieved through alterations to SysML Sequence Diagrams. Because Sequence Diagrams explicitly represent what the system is doing, the ordering of activities, and the allocation of those activities to physical elements (or blocks) it is easy to define the external environment as a physical element that is checked at the beginning of each sequence and

alters the properties of each subsequent activity within that sequence. Figure 53 provides a visual representation of this type of addition to the Detect Mines activity.

Figure 53 Integration of Impactful Environmental Variable in Subsequent MBSE MEASA Iteration (Inclusion of Environment as First Event in Sequence Diagram)



Note that the Minefield Environment is now included as a Physical Entity that activities may be allocated to within the Sequence Diagram. In this example a new activity "Check Environmental Conditions" has been added and is allocated to the environment. The activity produces an "Environmental Conditions Impact" that is used as the trigger to the first activity in the sequence (note in this case the Sequence Diagram actually represents two alternative loops so there are two potential first activities in the sequence). Definition of the environment as a Physical Entity is primary enabler of

inclusion of impactful environmental variables in subsequent iterations of the MBSE MEASA. This allows for inclusion of activities that check the environmental conditions at the beginning of any number of activities, which ensures that each of the sub activities occur subject to any alterations to the environmental conditions. Note that this could be implemented by including a series of "if-then" statements before every potential activity, but this inclusion of the environment as a physical entity and the addition of an environmental checking activity allows the impact of environmental conditions to promulgate throughout an entire activity in a far more concise manner.

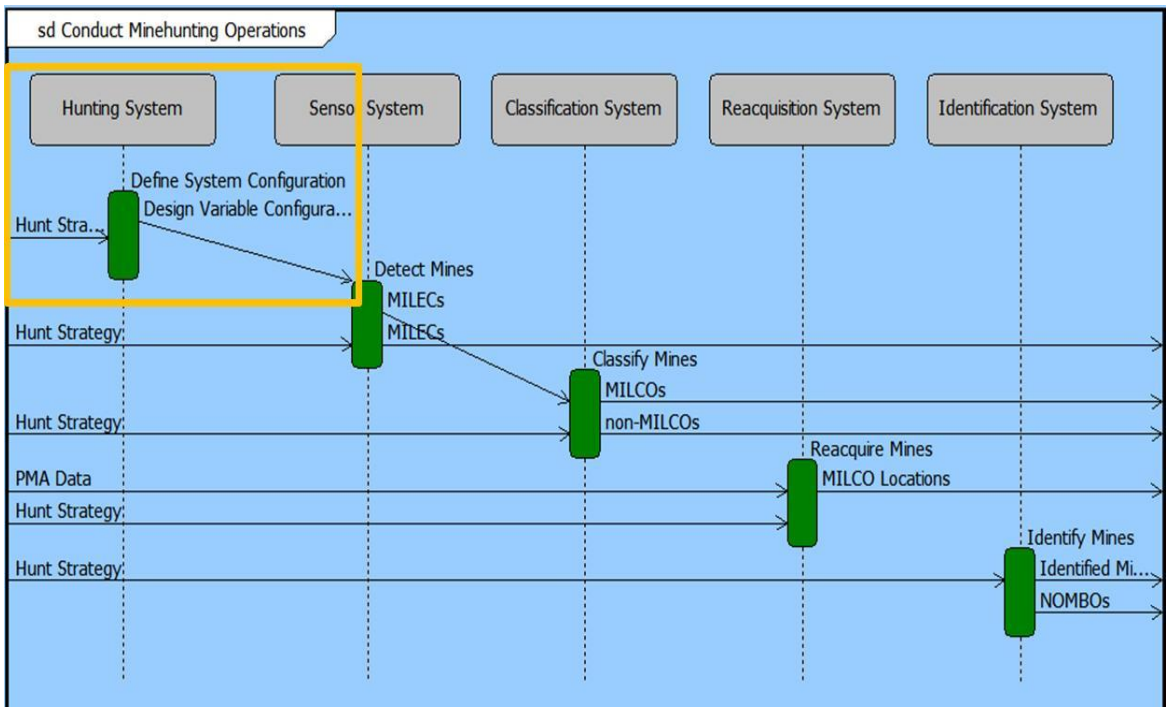
2. Iteration of MBSE MEASA for Significant In-Category Interactions

This section provides guidelines and illustrative examples for situations where analysis suggests that there is a potentially impactful interaction between variables within the same category. Case 2a describes situations where the impactful interactions occur between design variables, Case 2b describes situations where the impactful interactions occur between operational variables, and Case 2c describes situations where the impactful interactions occur between environmental variables. Note that an interaction between variables (of any type) suggests that the impact of an increase (or decrease) to the value of one variable is different depending on the value of another variable. For instance, the impact of an increase to the Probability of Detection of an MCM system may be different depending on the number of passes that the system conducts through the minefield (this corresponds to Case 3a, an interaction between a design variable, the Probability of Detection, and an operational variable, the number of minefield passes). If the Probability of Detection is set to some minimum value, additional minefield passes may be required. Similarly, if the Probability of Detection is set to a maximum value, fewer minefield passes may be required. The relationship can also be considered in the opposite direction, where the ability to conduct a given number of minefield passes may necessitate a certain probability of detection. The purpose of Cases 2a, 2b, 2c, 3a, 3b, and 3c is to provide guidelines regarding the integration of these types of analysis results from one iteration of the MBSE MEASA into future iterations of the MBSE MEASA.

a. Iteration of MBSE MEASA for Impactful Interactions between Design Variables

Recall that iteration of the MBSE MEASA when individual design variables are identified as potentially impactful requires alterations to existing Requirement Diagrams. These alterations were straightforward and did not require alterations to any other SysML products. Iteration of the MBSE MEASA when interactions exist between design variables requires additional work. For example, consider an analysis result that suggests that there is an impactful interaction between the probability of detection of an MCM system and the maximum search speed of the MCM system. This cannot be implemented within SysML through a simple alteration to the Requirement Diagram because the appropriate probability of detection is now dependent on the maximum search speed (and vice versa). Further, this cannot be implemented in SysML through alterations to either the Detect Mines activity or the Intra Minefield Transit activity because analysis results that identify potentially impactful interactions are not based on any assumptions of sequence (the user cannot simply assume that the search speed can be set and the probability of detection can be altered through an "if-then" statement simply because the transit activity occurs first because the interaction may imply that a reduced maximum search speed is sufficient provided the system has an increased probability of detection). Accordingly, integration of impactful interactions between design variables requires a user to alter the system operation at a level of abstraction that includes both of the design variables of interest. In this case, this means that the SysML products must be altered for the complete Minehunting sequence, rather than the specific activities associated with Mine Detection (where the Probability of Detection and the Maximum Search Speed could be altered directly if there were no interaction between those variables). Figure 54 provides a visualization of the revised Sequence Diagram for Minehunting.

Figure 54 Integration of Impactful Interactions Between Design Variables



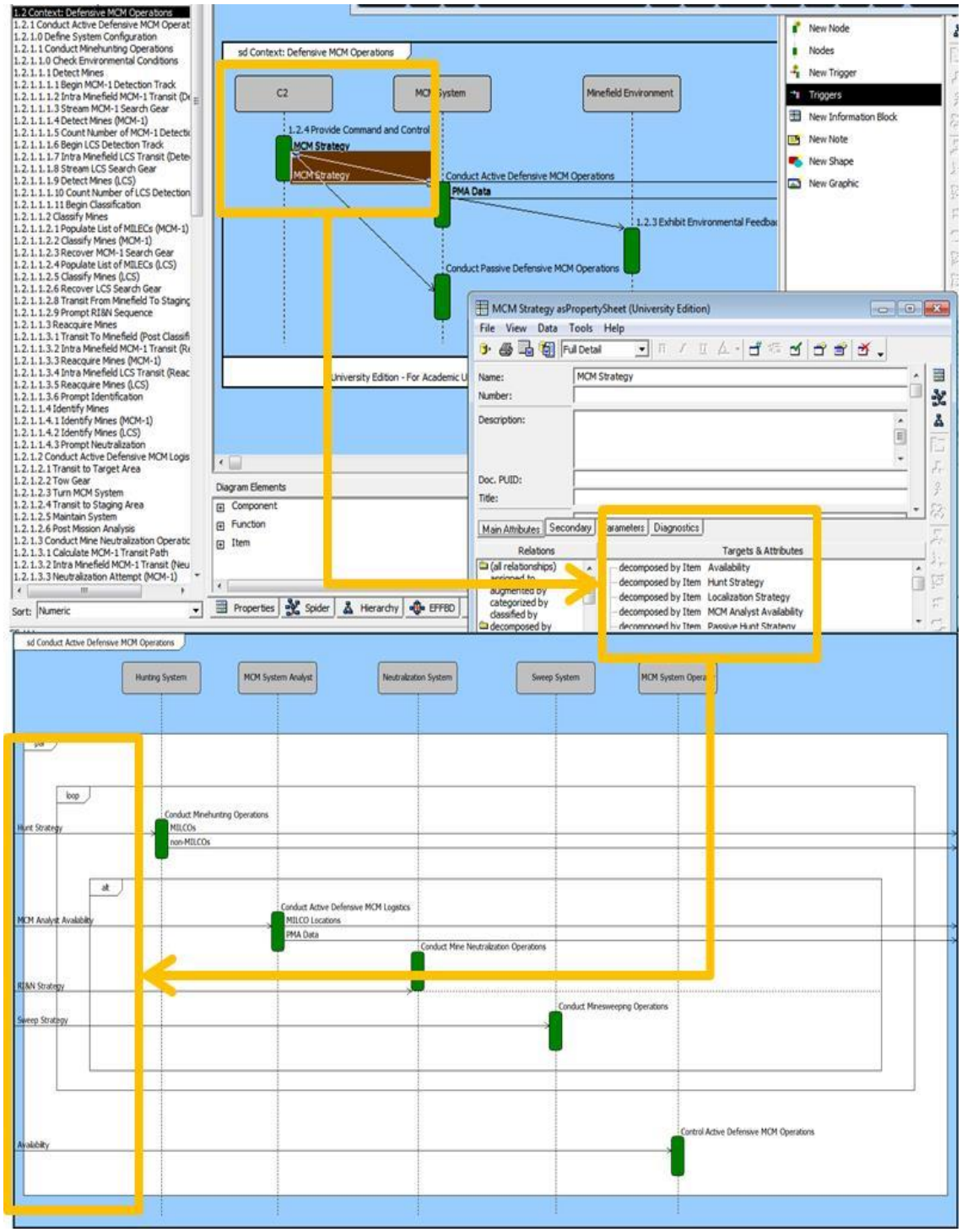
Note that an additional activity is now included in the Minehunting Sequence Diagram. The previous version of the Sequence Diagram did not include an initial activity used to define appropriate system configuration. This activity specifies appropriate values for each of the activities within Detect Mines. This is preferable to supplementing the Detect Mines with a series of "if-then" statements (while it would be possible to add a series of these statements for every design variable interaction it could become untenable if there were a simple number of design variable interactions).

b. Iteration of MBSE MEASA for Impactful Interactions between Operational Variables

Case 2b describes scenarios where analysis results suggest that there are impactful interactions between operational variables. Once again, this cannot be implemented within SysML through straightforward alterations to Requirement Diagrams or through additional "satisfied by" relationships within Requirement Diagrams. As with Case 2a, impactful interactions between operational variables requires the user to consider the

system at a level of abstraction above the variables that the analysis has identified as having an impactful interaction. As an example, consider an analysis result that suggests that there is an impactful interaction between the number of passes that the system conducts through the minefield and the percentage of the minefield that is searched by surface assets (rather than airborne assets). As with the Case 2a, there is no sequence implied by the analysis result that there is an interaction between these variables (a user cannot just set the number of minefield passes and subsequently select a preferred minefield search percentage). Accordingly, the impact of the interaction between the operational variables must be incorporated at a higher level of abstraction. It may also be useful to establish an external "Command and Control (C2)" physical entity that manages each of the operational decisions (this is shown in Figure 55). In the example the C2 entity is responsible for the Provide Command and Control activity, which specifies an MCM Strategy for Active Defensive MCM Operations. This MCM Strategy is decomposed into a Hunt Strategy, Localization Strategy, etc., which is then used as an input to each of the sub activities to Active Defensive MCM Operations. In this case, the C2 specifies a broader MCM Strategy, which includes the Hunt Strategy that is utilized for Mine Detection, which can describe the appropriate operational decisions regarding the number of minefield passes and the surface search percentage.

Figure 55 Integration of Impactful Interactions Between Operational Variables



c. Iteration of MBSE MEASA for Impactful Interactions between Environmental Variables

Case 2c describes situations where analysis results suggest that an interaction between environmental variables has a potential impact on system performance. While interactions between design and operational variables required substantially different strategies compared to situations where only a single variable impacted system performance the integration of interactions between environmental variables closely mirrors the integration of a single impactful environmental variable. Recall that Case 1c advocated the definition of an environmental checking activity at the beginning of any activity sequence where analysis indicated that an environmental variable impacted system performance. If multiple environmental variables impact system performance the same strategy may be used because the introduction of the environmental checking activity at the beginning of the sequence ensures that the outputs may be directed to any subsequent activity and ensures that the result of the activity promulgates throughout the entire sequence. For example, if there is an interaction between the impact of sea state and the impact of current (or drift) conditions, both of these may be included in the Check Environmental Conditions activity (just as in Case 1c) and the output can inform any associated subsequent activity.

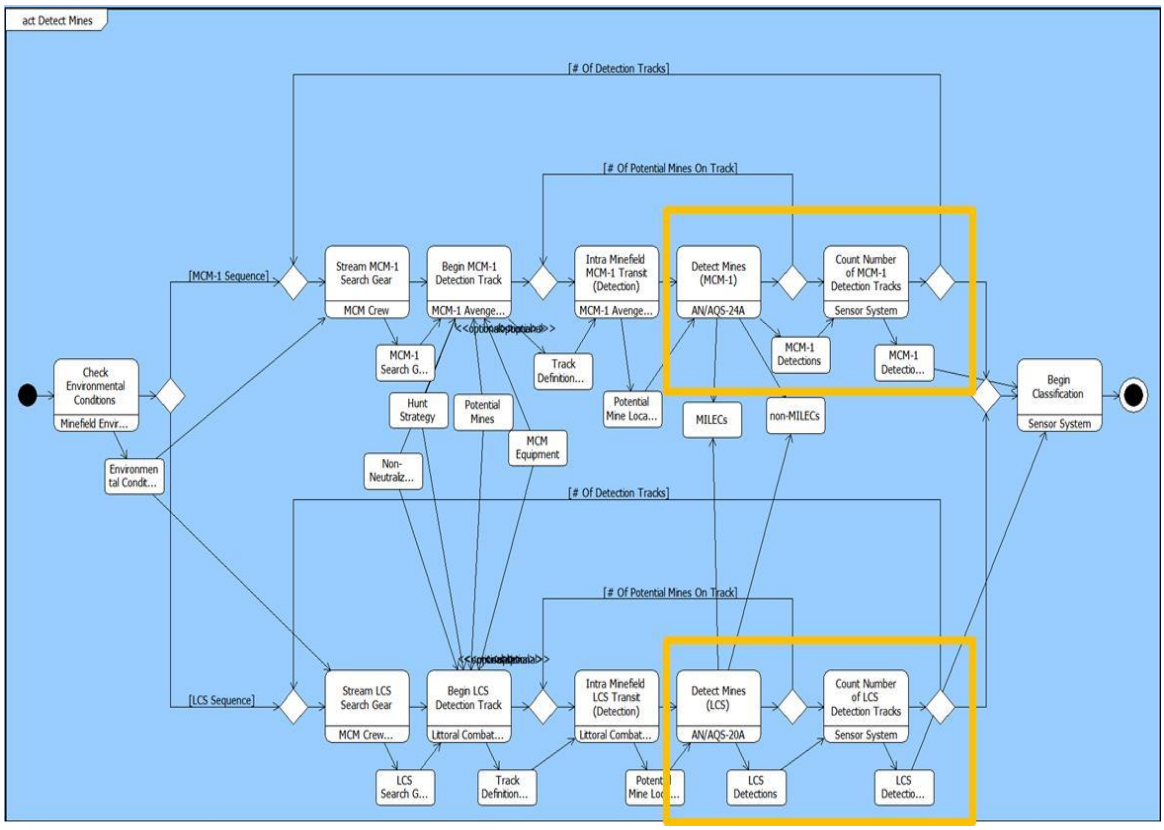
3. Iteration of MBSE MEASA for Significant Between Category Interactions

This section discusses situations where analysis suggests that there are potentially impactful interactions between different variable types. Case 3a describes situations where the impactful interactions occur between design variables and operational variables, Case 3b describes situations where the impactful interactions occur between operational variables and environmental variables, and Case 3c describes situations where the impactful interactions occur between environmental variables and design variables.

a. Iteration of MBSE MEASA for Impactful Interactions between Design Variables and Operational Variables

Case 3a describes scenarios where analysis indicates that an interaction between and design and operational variable impacts system performance. Continuing the variables used in previous examples, this example utilizes the probability of detection as the design variable and the number of passes through the minefield as the operational variable. To ensure maximum applicability of the MBSE MEASA, note that the sequencing should not be assumed. Even though the focus is the design of the system, and therefore on the definition of design variable values, it is imprudent to design a system that can only operate in certain operating systems. Likewise, it is unrealistic to assume that, based on a given operational decision; it will be possible to alter the value of a design variable. However, there are numerous mechanisms within SysML to ensure that there is a process for including the interaction in both possible directions. In Figure 56 the activity for Detect Mines directly precedes the activity for Count Number of Detection Tracks. The Detect Mines activity produces an item (MCM Detections) that directly informs the activity for Count Number of Detection Tracks. Note that (as developed in Case 1b) this activity diagram incorporates the Hunt Strategy on the left of Figure 56. This allows a user to specify the appropriate number of detection tracks that will be conducted, which either can be held constant or updated based on the input from the Detect Mines activity. Furthermore, because the Hunt Strategy is inputted at the beginning of the activity sequence, it can update the procedure for Detect Mines. The inclusion of a an operational consideration at the beginning of an activity sequence (in this case, the Hunt Strategy, as developed in Case 1b) and the direct linkage of the design variable to the operational variable allows a user to account for any interactions between the variables.

Figure 56 Integration of Impactful Interactions Between Design and Operational Variables

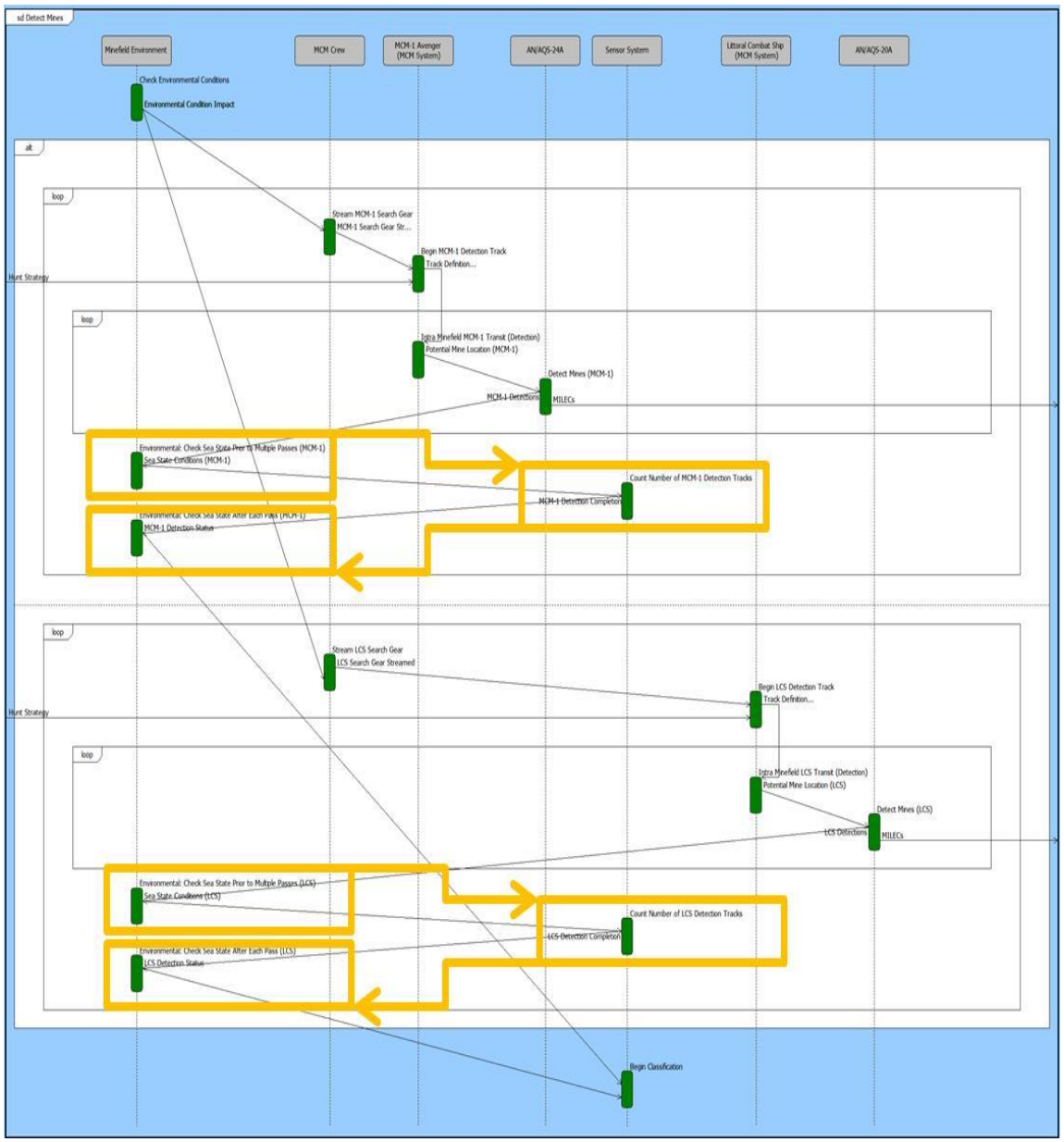


b. Iteration of MBSE MEASA for Impactful Interactions between Operational Variables and Environmental Variables

Case 3b describes situations where there is an impactful interaction between operational and environmental variables. Recall that the integration of impactful environmental variables focused on the addition of the system environment as a physical entity that performed an activity that provided environmental conditions to each of the subsequent activities in a sequence. A similar technique is valuable in Case 3b, however additional work is necessary. As emphasized, it is inappropriate to assume sequencing when updating SysML products based on analysis results that suggest impactful interactions. In Case 3b this is particularly important, since environmental conditions can impact system operation and system operation can impact environmental conditions. The definition of the system environment as a physical entity within the system allows a user

to model each of these potential situations. Figure 58 provides an example of a supplemented sequence diagram for mine detection where prior analysis suggested that the interaction between the sea state and the number of minefield passes has an impact on system performance. Note that the environmental condition (in this case the sea state) is checked before and after the operational decision activity (the decision on the number of minefield passes). This allows a user to specify before the operational decision any alterations that should be made based on the environmental condition, and also allows the user to update the environmental condition based on changes to the operational decision. Note that Figure 57 demonstrates this alteration for both the MCM-1 and the LCS sequences.

Figure 57 Integration of Impactful Interactions Between Operational and Environmental Variables

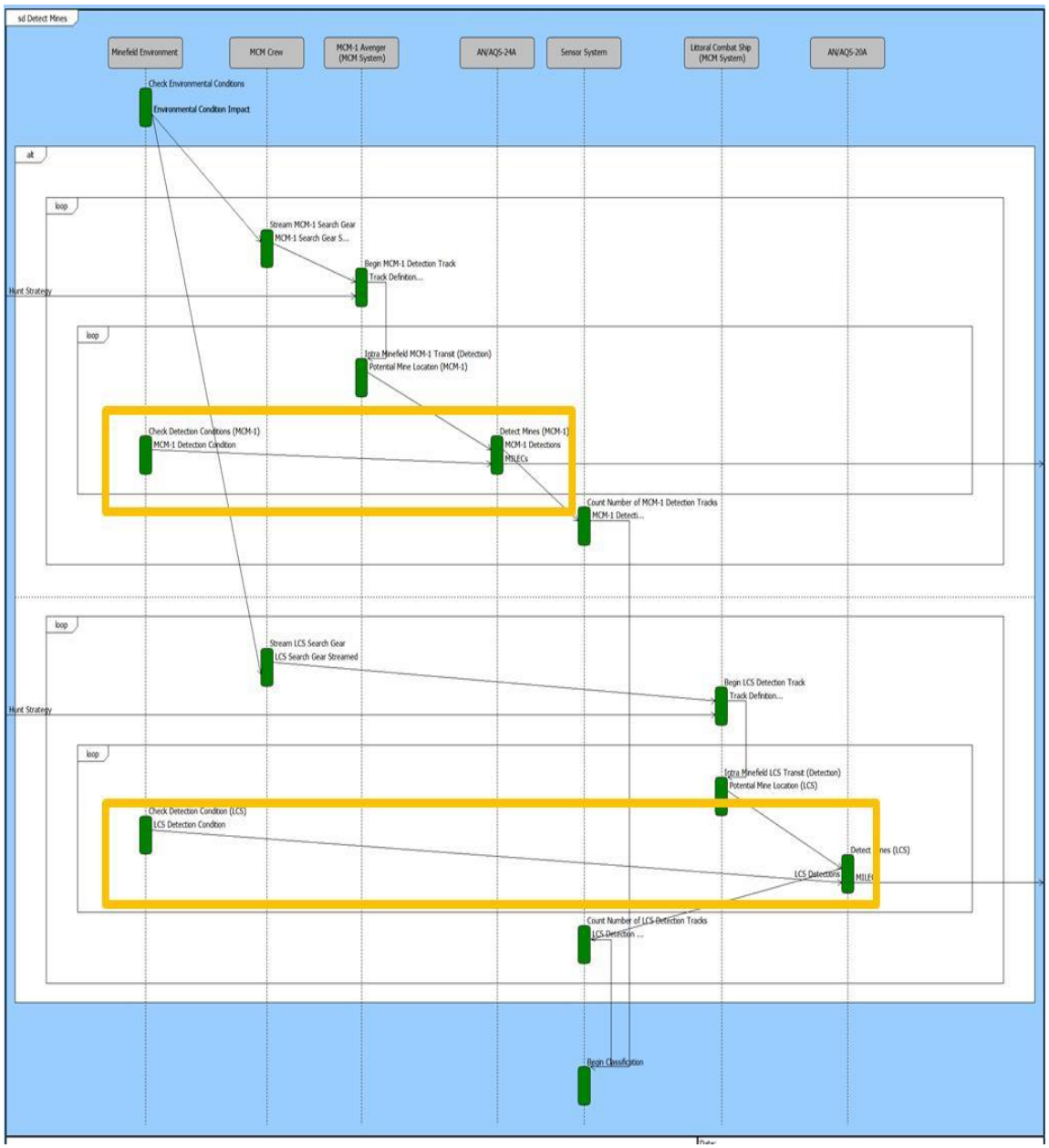


c. Iteration of MBSE MEASA for Impactful Interactions between Environmental Variables and Design Variables

The integration of analysis results that suggest that there is an impactful interaction between environmental and design variables (Case 3c) is less cumbersome than Case 3b. While it is possible to modify the system operational implementation

continuously based on environmental conditions, the system design typically cannot be continuously modified throughout a simulation model (or a real life operation) to suit altered environmental conditions. Accordingly, Case 3c can be implemented similarly to Case 1c, an activity should be added to any sequence where a potentially impactful interaction exists and the impact of that altered environmental condition should be incorporated within that activity. Because more detail is available in Case 3c scenarios than was available in Case 1c scenarios (the user knows specifically what design variable-environmental variable interactions impact system performance) it may be useful to have the environmental condition directly feed the design variable of interest (this should not make any difference in terms of the underlying SysML model but may aid communication to stakeholders). Figure 58 provides an example within a mine detection sequence where the sea state is modeled as an environmental condition that directly feeds the activity for mine detection (which models the design variable for the probability of detection).

Figure 58 Integration of Impactful Interactions Between Environmental and Design Variables



Systems engineering recognizes the importance of iteration. The need to feed subsequent processes based on past results is emphasized throughout the systems engineering literature. However, the MBSE MEASA goes beyond the simple acknowledgment that iteration is important. The MBSE MEASA considers a broad range

of potential variables (Design, Operational, and Environmental). The MBSE MEASA also recognizes that interactions between these variables are inevitable and likely to have potential impacts on system performance. Accordingly, the MBSE MEASA provides guidelines and illustrative examples for iteration of the methodology. These guidelines and examples should allow any user who has followed the MBSE MEASA and developed SysML architecture products, constructed a simulation model, and conducted analysis of the modeling results to update the previously developed SysML architecture products for a wide range of potential analysis results.

IV. MBSE MEASA DEMONSTRATION AND ANALYSIS

In order to highlight the expected utility and applicability of the MBSE MEASA, this research presents an analysis comparing the operational effectiveness of future and current U.S. Navy mine warfare systems. As mentioned, this analysis leverages an operational simulation developed by Becker et al. (2014). The focus of the original research was comparing the performance of future U.S. mine warfare capabilities (evaluated through a simulation of the LCS in a mine warfare operation) against current U.S. mine warfare capabilities (evaluated through a simulation of the MCM-1 in a mine warfare operation). The research focused on the ability of both systems to clear a minefield in a representative operational scenario where the MCM systems began the operation at the potential minefield and each system was only capable of making one pass through the minefield. This demonstration adds four additional variables to the earlier investigation, specifically considering the need for each system to transit to the minefield prior to commencement of minehunting activities and examining the impact of making multiple passes with each system within the minefield. Note that, as for most detailed analyses, the analysis of system performance may be highly dependent on the established initial conditions. A major advantage of the MBSE MEASA is the ability to capture these initial conditions in a standardized set of SysML products, which can be presented to stakeholders to determine relevance, operational feasibility, and correctness and can also be rapidly updated to reflect any alterations to guidance or stakeholder preference. As an important note, this demonstration will focus primarily on the development of an operational simulation model (rather than synthesis models) and will only present a single iteration of the methodology. This should not understate the importance of iteration, recall that Chapter III presented illustrations of the iteration procedure for the full range of potential analysis results.

A. SYSTEM DEFINITION AND SYSML PRODUCT GENERATION

As mentioned in Chapter III, this research focuses on Active, Defensive minehunting and neutralization operations for influence mines between 40–200 feet below the surface. After the system enters the minefield, those operations are typically conducted by a linear sequence of activities, namely: Mine Detection, Mine

Classification, Mine Identification, and Mine Neutralization. Based on the MBSE MEASA, a set of system requirements describes what a system must do in terms of each of these activities. Furthermore, based on the description of the full set of MCM challenges outlined NWP 3–15 and PEO LMW Instruction 3370.1A, the additional activity of transit to the minefield is included in this analysis.

Analysis of that MCM doctrine dictates the functions that satisfy each system requirement. These functions then define an operational model of minehunting and neutralization operations. The system architecture also defines the physical elements that satisfy those requirements, which supports the operational model as well as any synthesis model (cost, physical, etc.) of the system. Simulation and analysis of these models describes the set of systems that best satisfy the initial set of requirements. Iteration of the process evaluates those requirements in more detail.

1. Requirements Analysis

As demonstrated in Chapter III, the MBSE MEASA begins with creation of a SysML Requirement Diagram. Chapter III presented an example Requirement Diagram that detailed the system requirements for the minehunting capability of the MCM system. A similar diagram (Figure 59) presents a SysML Requirement Diagram for the additional logistics requirements that exist for a MCM system.

Figure 59 SysML Requirement Diagram: Perform Logistics Functions

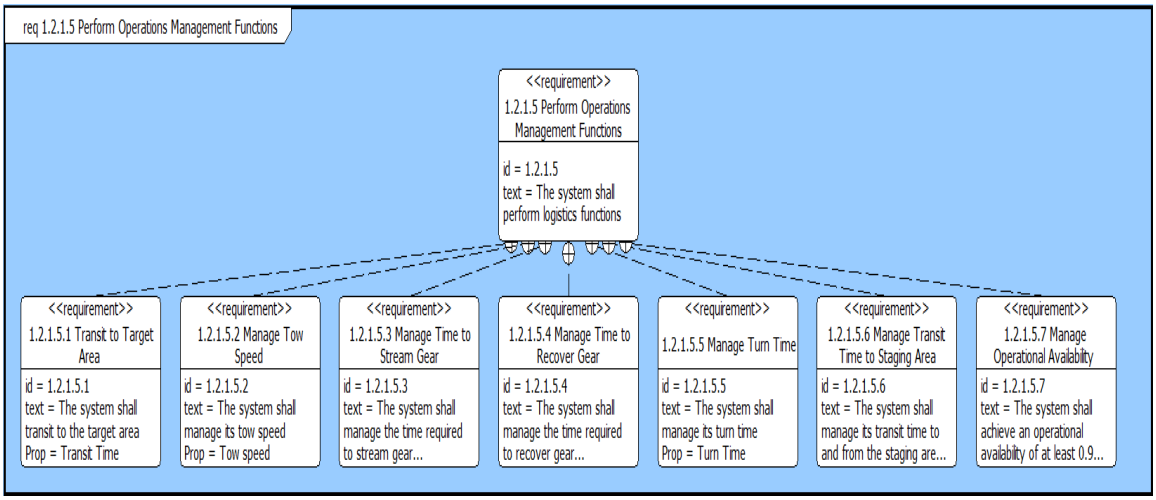


Figure 59 presents an overview of the requirements associated with system operations management. System operational models must represent these requirements. Specifically, operational models must represent (and potentially vary) transit to the target area. Furthermore, a tow speed must be modeled, streaming of the search and neutralization gear must be modeled (as a note, "streaming" is defined as the time to deploy MCM equipment prior to entering a minefield), recovery of the search and neutralization gear must be modeled, a turnaround time must be modeled once the system reaches the edge of the minefield, transit from the staging area to the minefield must be modeled, and operational availability must also be modeled. Each of these operations management requirements will be incorporated as variables into the external simulation model to determine their impact on the overall system effectiveness. Functional Architecture products that capture the behaviors necessary to support these requirements (and the requirements for Mine Hunting, presented in Chapter III) must provide additional detail regarding the representation of these requirements in the simulation.

2. Functional Architecture

As presented in Chapter III, the purpose of Functional Architecture development is to describe the system of interest in terms of how it will satisfy the previously defined set of system requirements. This prompts development of Activity Diagrams (which present not only activities, but also external objects that trigger each activity), Sequence Diagrams (which defines the ordering of system activities as well as the interactions between system objects), Use Case Diagrams (which describes the set of actors that conduct each activity, as well as potential extensions of each activity), and State Machine Diagrams (which describe how alterations to system operating conditions alter the implementation of each activity). Note that it is necessary to define some of the physical elements that comprise a system (as well as external physical elements) but all effort should be made to remain as solution neutral as possible during the creation of functional architecture products to ensure that the range of potential solutions are not unnecessarily restricted. For example, in the case of the MCM-1 Avenger, the exact physical system that will conduct Airborne Mine Detection is known (the AN/AQS-24A) because the system has already been built. However, as a general rule for systems that have not

already been built, this level of detail should not be included until physical architecture products are developed. Within the functional architecture it is sufficient (and preferable) to describe that element as a Sensor System. Given that these SysML products is intended to be used in conjunction with the SysML products developed in the previous chapter to define a discrete event simulation for an Active, Defensive MCM system, the functional architecture products that are most relevant are Activity Diagrams. Recall that Active, Defensive MCM operations are defined by a discrete sequence of: transit to minefield, detect mines, classify mines, reacquire mines, identify mines, and neutralize mines. In Chapter III the high level Activity Diagram defined that decomposition from Active, Defensive MCM Operations to Minehunting Operations to Detect Mines was shown, but did not provide sufficient detail regarding Mine Classification, Reacquisition, Identification, and Neutralization to enable development of a simulation model. Figure 60, Figure 61, and Figure 62 present those activities in more detail to guide this development.

Figure 60 Activity Diagram (Classify Mines)

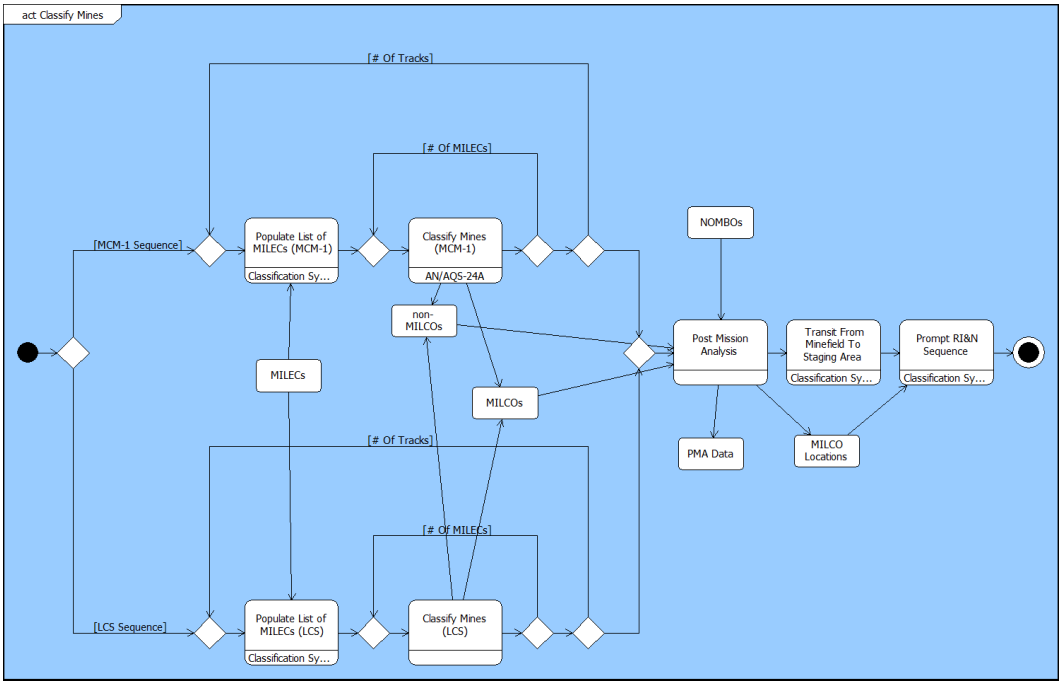


Figure 61 Activity Diagrams (Reacquire Mines & Identify Mines)

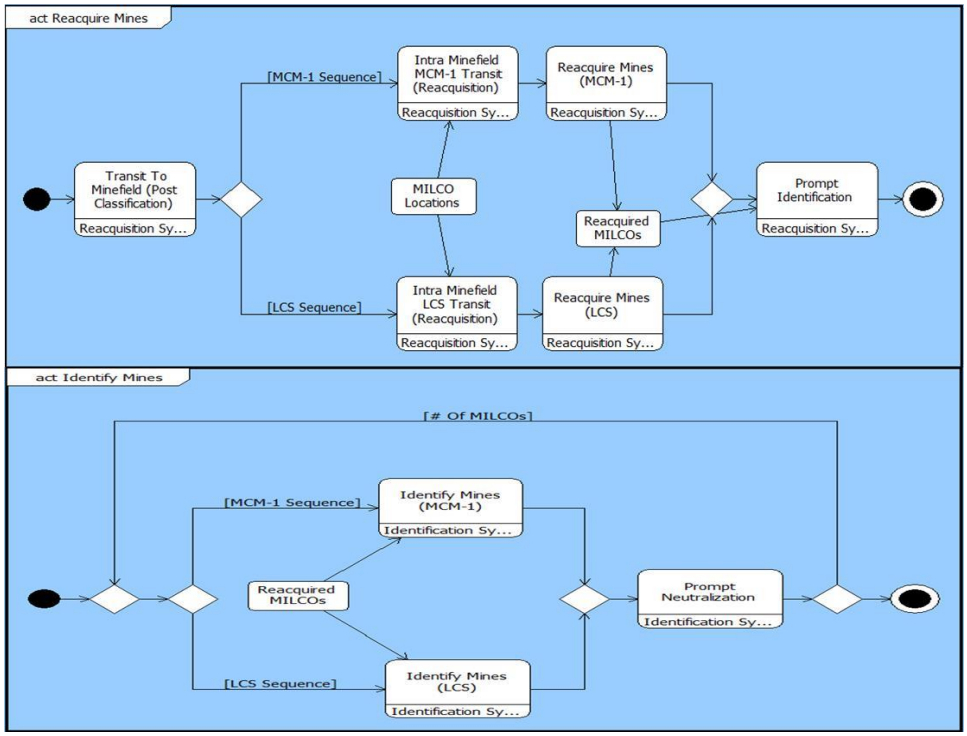
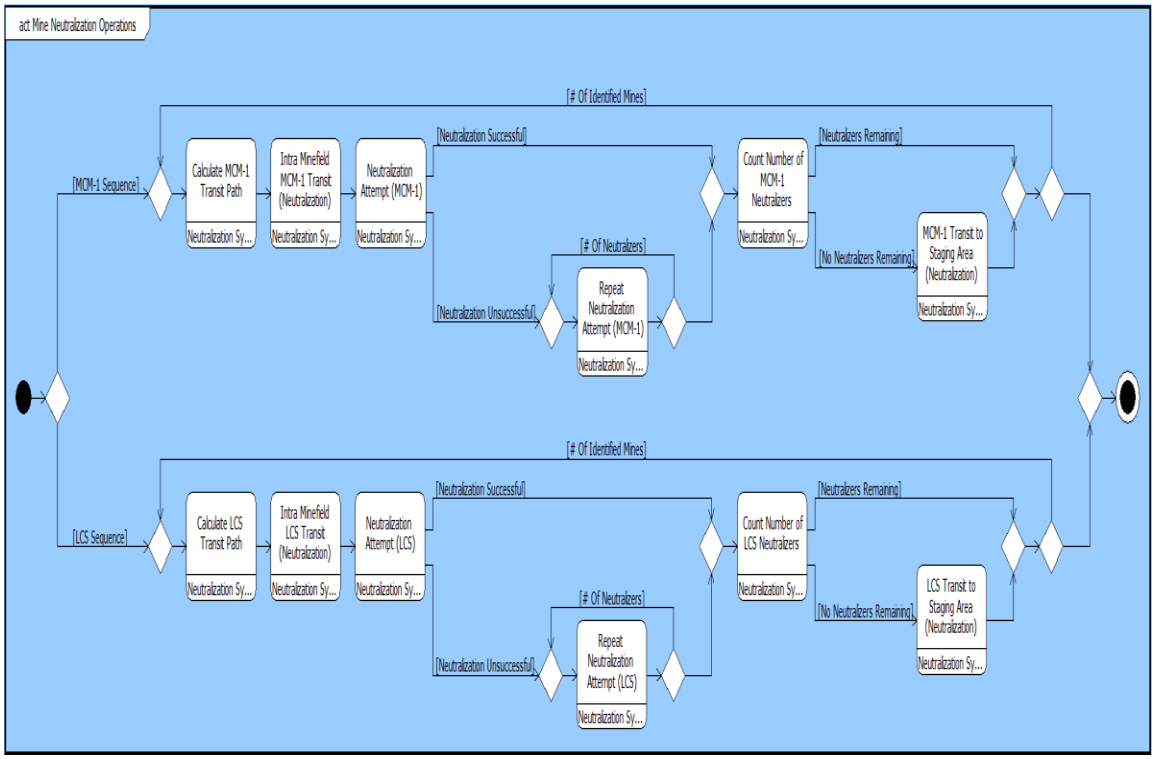


Figure 62 Activity Diagram (Neutralize Mines)



Note that while this research focuses on presentation of Activity Diagrams, the Sequence Diagram and State Machine Diagram presented in Chapter III specified the ordering of the Activity Diagrams within the discrete event simulation (as a practical note, it is far easier in the CORE software to map physical components to functions within Sequence Diagrams than within Activity Diagrams). The Use Case Diagram was necessary to establish the actors that performed each high level activity, but was less vital to the development of the external simulation. Note that Use Case Diagrams are often vitally important to systems where multiple missions must be defined and exercised, as would be the case if this research were expanded beyond Active, Defensive MCM operations to include other mine warfare operations, as detailed previously.

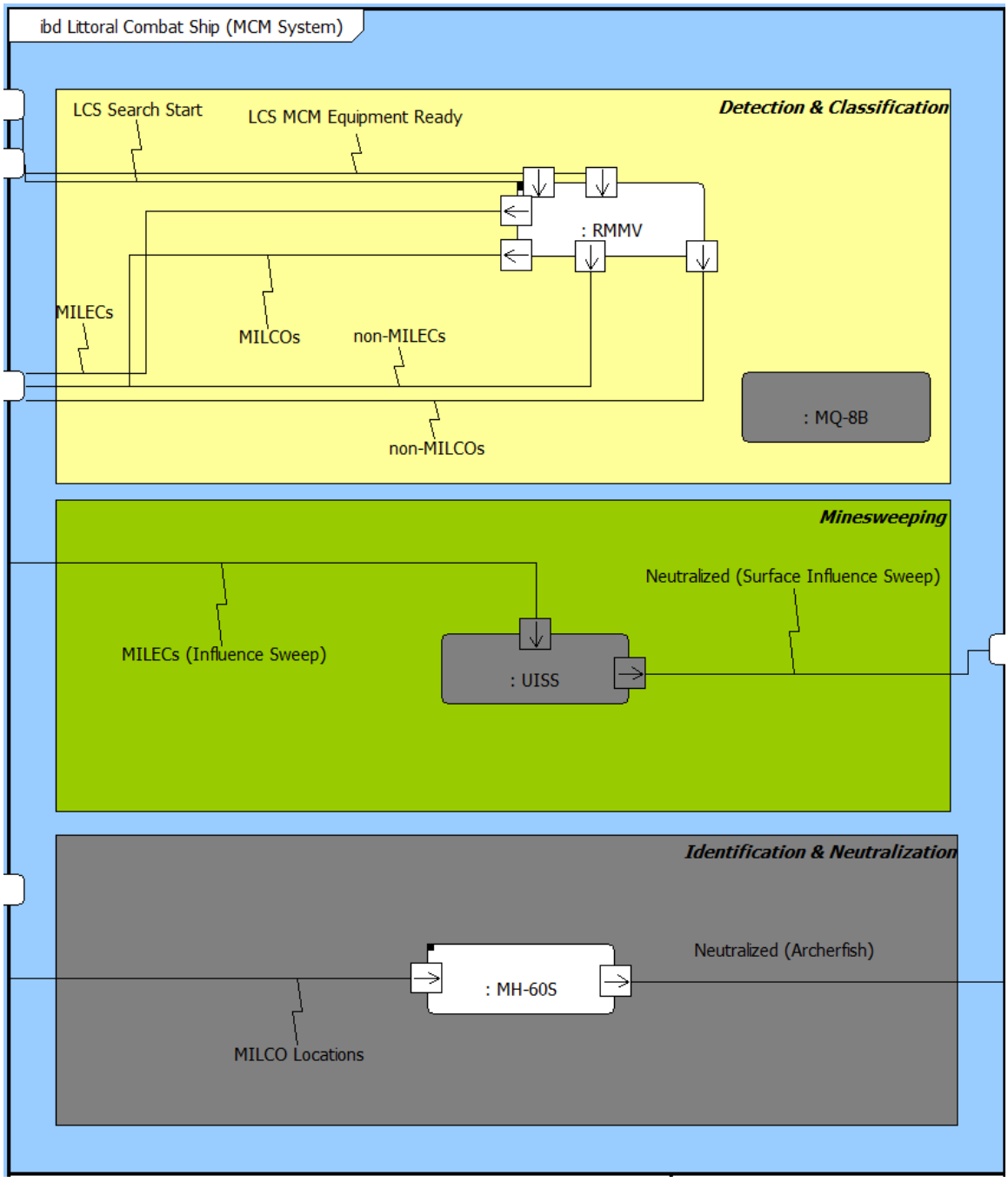
3. Physical Architecture

As prescribed by the MBSE MEASA, completion of the set of Functional Architecture products (specifically the Activity, Sequence, Use Case, and State Machine Diagrams) prompts the development of Physical Architecture products (Block Definition and Internal Block Diagrams). In this case, Chapter III presents a comprehensive Block Definition Diagram (Figure 45). As noted, definition of the Block Definition Diagram terminated at the system level, for systems that are less well defined it may be necessary to expand Block Definition Diagrams to include subsystems, system components, and system end items. As mentioned, physical architecture definition should proceed to a sufficient level to develop a model or simulation of the system of interest, which requires that the each system function can be allocated to one or more system components. While this is not evident from an isolated study of the Block Definition Diagram, considering the diagram simultaneously with the functional architecture products (which describe system components in a solution neutral form) as well as Internal Block Diagrams confirms that each system function is allocated to appropriate system components.

Figure 46 presented an Internal Block Diagram for the MH-53E, which showed that the MH-53E was capable of completing the full detection through neutralization sequence of Active, Defensive MCM operations (albeit with a required change to the supporting subsystems) for MCM-1 configurations. Figure 63 presents an Internal Block

Diagrams for the LCS MCM system. The Internal Block Diagrams shows that it is necessary for the LCS to employ multiple systems to complete the full sequence of mine detection through neutralization. Specifically, the RMMV completes mine detection and classification while the MH-60S completes mine identification and neutralization (note that a further decomposition of the Internal Block Diagram focused solely on the MH-60S would suggest that the MH-60S requires a supporting external system to conduct mine neutralization, which in this case is assumed to be the AN/AQS-25 Archerfish system (this level of detail was shown in Figure 45). Note that organizational blocks are once again shown to aid visualization; these organizational blocks are not necessary elements of Internal Block Diagrams but are shown to demonstrate the segmenting of physical systems that conduct each mine warfare activity. Those blocks shown in grey (the MQ-8B Fire Scout and the Unmanned Influence Sweep System) are unrealized systems (at least in terms of utilization for mine countermeasures operations) expected to provide future functionality that are beyond the scope of this study but are included for completeness and to facilitate better comparison with the Internal Block Diagrams shown for the MCM-1 Avenger configurations.

Figure 63 Internal Block Diagram (LCS MCM Systems)



It is certainly difficult to provide a complete demonstration of the utility of SysML architecture products through presentation of static figures. While these figures do provide a defined picture of the major functional and physical properties of potential systems, it is difficult to present the level of detail associated with the connections

between each of the diagrams. Much of the value of a coordinated set of architecture products is that changes to system requirements, functions, elements, etc., in one diagram will promulgate through each diagram, substantially reducing the need for rework and providing nearly instantaneous checks on consistency between the functional and physical representations of a system. While this is certainly a limitation of architecture presentation through static figures, a sufficient level of detail has been presented to guide development of an external model of Active, Defensive MCM operations based on the SysML architecture products. Recall that the focus of this chapter is to demonstrate the importance of aligning an external model or simulation with previously developed architecture products. Accordingly, the next section will provide an overview of the external model built in support of this research with a focus on ensuring that each of the functions and activities, as well as the appropriate physical elements, are represented properly in the simulation model.

B. MODEL DEFINITION

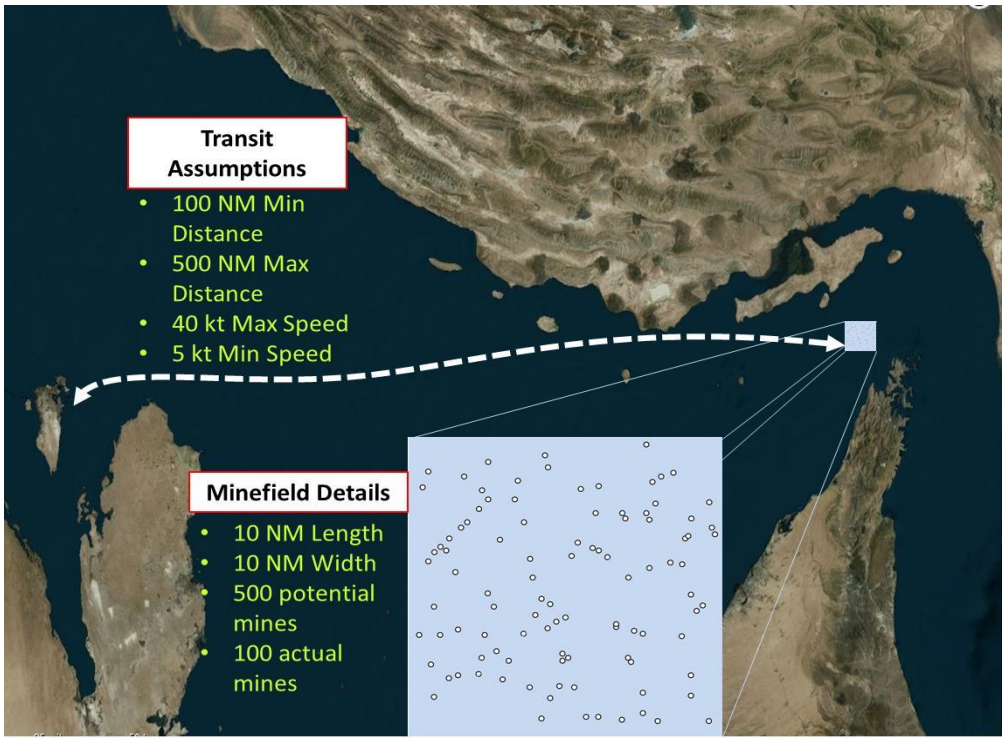
Examination of the set of functional and physical architecture products defined in the first three stages of the MBSE MEASA serve as the primary guidance for the development of an external model of Active, Defensive MCM operations. The functional architecture products have defined the set of behaviors that must be represented in the simulation and the physical architecture products have defined the set of systems and subsystems that must be represented in the simulation. As mentioned, the choice of simulation approach is often highly dependent on the expertise of the user, in this case familiarity with discrete event simulation, as well as a system of interest that performs a clearly defined sequence of events, led to the selection of a discrete event simulation to model Active, Defensive MCM operations.

1. Model Representation

Chapters III and IV presented the set of functional behaviors and activities that must be represented in the model using a set of SysML products. Similarly, another set of SysML products presents the systems and subsystems represented in the discrete event simulation. Detailed examination of those SysML products suggests that the discrete

event simulation must represent three distinct stages of operation: transit to and from the minefield, minehunting, and mine neutralization. Furthermore, physical systems must exist in the simulation to conduct transit, mine detection, mine classification, mine identification, and mine neutralization. The SysML products are the basis for model construction; however additional clarification may be required for the reader unfamiliar with SysML products or discrete event models. Appendix D presents a mapping of SysML products to the external simulation. To aid with description of these process and the related physical systems within the discrete event simulation, Figure 64, Figure 65, Figure 66, and Figure 67 provide a visual representation of each stage of operation.

Figure 64 Transit to the Minefield and Minefield Definition



As prescribed by the system architecture, the operation begins with transit to the minefield. Note that while Figure 64 shows a notional operational environment, the simulation varies the total transit distances and transit speed within the boundary conditions specified by Figure 64 (increased detail will also be shown in Table 8 and Table 9) to ensure that the results are as generalizable as possible. To facilitate

comparisons between system alternatives the characteristics of the minefield are constant. The simulation creates the minefield by assigning a random x and y coordinate to 400 non-mines and 100 mines, which are the entities within the discrete event simulation. Note that this constant specification of the minefield means that the application of the analysis results should be restricted to similar operational scenarios. In this case, the specific operational scenario was chosen after discussion with subject matter experts suggested that this was a stressing implementation of a likely operational scenario. That said, it is important to note that, as demonstrated by Allen, Buss, and Sanchez (2004), environmental factors such as current speed, current offset, and range from the sensor to a mine may also have a substantial impact on system performance. The simulation does not include these factors, but investigation is possible in future work. Several other assumptions and limitations, as presented in Becker, et al. (2014) may be of interest and may restrict the applicability of the results, particularly:

1. The only mines present would be bottom mines in water deeper than 200 feet (Becker 2014, 154)
2. Sea state, weather, water visibility, and sea floor type were not modeled
3. Each target is only considered a single time (the sensor is modeled as a "cookie cutter" sensor). This is based on SME opinion that the search speed is slow enough and the tracks are spaced closely enough that each target can be detected in a single instance and that any target that is missed can be ignored

The system then transits to a staging area located several miles from the minefield (the distance from the staging area to the minefield as well as the transit speed from the staging area to the minefield are varied). After minefield definition and transit to the minefield is complete, the simulation moves to the detection function. Note that to this point the simulation models the MCM-1 configurations and the LCS simulations exactly the same; however, each configuration is represented differently in the discrete event simulation after this point due to the variations in the physical entities that conduct mine detection, classification, reacquisition, identification, and neutralization operations. Specifically, MCM-1 configurations utilize multiple systems (the MCM-1 Avenger and the MH-53E) to conduct each stage of the operation while the LCS configurations utilize one system (the RMS) to conduct mine detection and classification and a second system (the MH-60S) to conduct mine reacquisition, identification, and neutralization. Figure 65

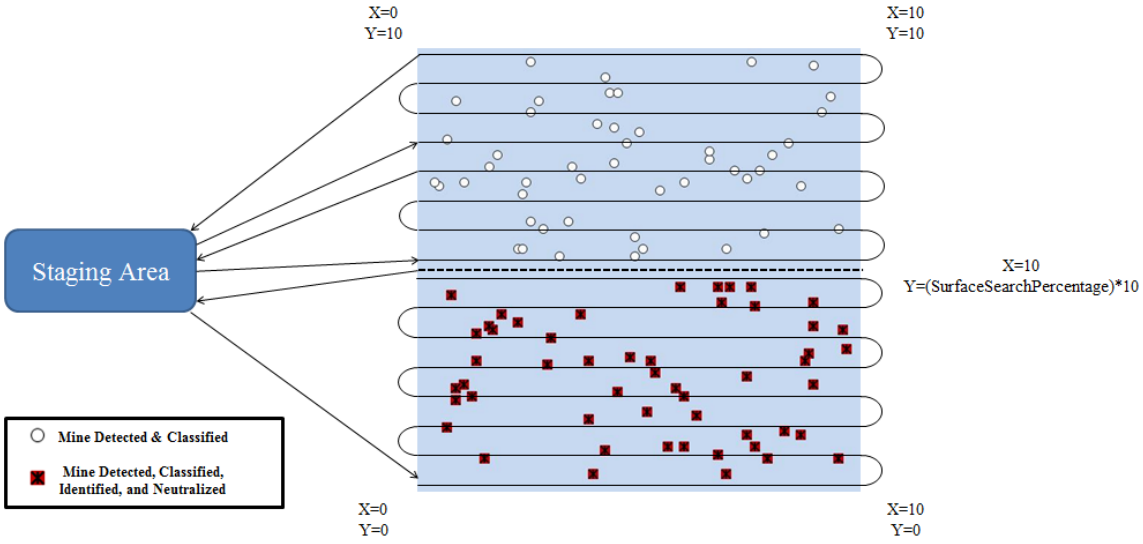
and Figure 66 provide visual representations of the simulation implementation of the MCM-1 configurations, while Figure 67 provides a visual representation of the simulation implementation of the LCS configurations. As mentioned, Appendix D presents a more detailed representation of each of those figures within the discrete event modeling software (ExtendSim).

Prior to the commencement of mine detection, the simulation further defines the minefield by varying the portion of the minefield that will be searched by the MCM-1 Avenger and the portion that will be searched by the MH-53E (note that this will not be necessary for the LCS configurations, as only one system performs mine detection). This is highlighted in Figure 65 (which assumes a that half the minefield is searched by the MCM-1 Avenger and half the minefield is searched by the MH-53E) where the y-coordinate on the right side of the figure is specified as the "Surface Search Percentage," and is later varied from 0.30 to 0.70.

Figure 38 and Figure 60, which presented Activity Diagrams for Mine Detection and Mine Classification, are the basis for the set of events defined in the simulation. In general, one or more simulation variables are associated with each event. Note that there is no system movement for the MCM-1 configurations between mine detection and mine classification; therefore Figure 65 presents detection and classification happening at each point in the minefield. The simulation implements a sequence where each search system, proceeds from left to right along a track, stopping at each potential mine and identifying it as either a MILEC or a non-MILEC. The simulation models detection of each potential mine, and proceeds to mine classification for those potential mines identified as MILECs. The MILECs are then classified as either MILCOs or non-MILCOs, the list of which is then saved for PMA. In the portion of the minefield being covered by the MCM-1 Avenger, the system proceeds with mine neutralization. In Figure 65, the lower half of the region is searched by the MCM-1 Avenger, which completes the full sequence of detection through neutralization, and the top half of the region is searched by the MH-53E, which completes only mine detection and mine classification. Note that in the example shown, the MCM-1 Avenger only requires a single sortie but the MH-53E returns to the staging area, because the search sequence requires two sorties to complete

(the simulation model varies the Sortie Time and it is different for each simulation run). As mentioned, one or more variables are associated with each event, the variables associated with mine detection and classification are: the search speed, the probability of mine detection, the probability of correct classification (for both MILCOs and non-MILCOs), the time to stream and recover the search gear, the number of tracks the system will complete per nautical mile, the time to turn around to begin a new track, the duration of each mine detection sortie, and the maintenance time required at the end of each sortie.

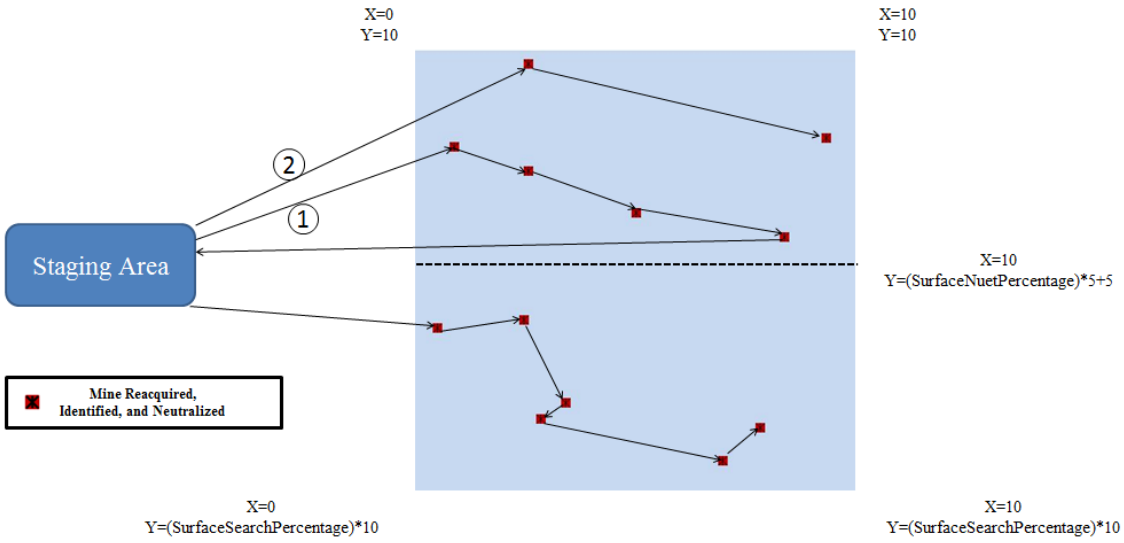
Figure 65 Detection and Classification: MCM-1 Configurations



After PMA has created a list of the MILCOs that must be reacquired for neutralization, both the MH-53E and the MCM-1 Avenger proceed to travel to each target and conduct a sequence of reacquisition, identification, and neutralization. Again, the percentage of the targets engaged by each system is varied. Each system is assigned a percentage of the MILCOs to neutralize, and a nearest neighbor algorithm dictates the sequence of MILCOs engaged by each system. If either the MH-53E or MCM-1 Avenger is required to return to the staging area during this portion of the simulation (due to the number of neutralizers carried on the system) the nearest neighbor algorithm resets, using

the staging area as the starting point (in Figure 66, this is shown for the top portion of the region, searched by the MH-53E, but not for the bottom portion of the region, searched by the MCM-1 Avenger). The full set variables associated with mine identification and neutralization are: the probability of reacquisition, the probability of identification (for both MILCOs and non-MILCOs), the probability of neutralization, the time to deploy and recover the reacquisition, identification, and neutralization (RI&N) gear, the time for reacquisition and identification (defined with both a mean and a standard deviation, assuming a normal distribution), the time for neutralization (defined with both a mean and a standard deviation, assuming a normal distribution), the speed of the neutralizers, the portion of the MILCOs neutralized by the MCM-1, and the portion of the MILCOs neutralized by the MH-53E.

Figure 66 Identification and Neutralization: MCM-1 Configurations



Recall that while each of the systems used in the MCM-1 configurations are capable of conducting the full sequence of mine detection through neutralization, the LCS configurations use two separate systems for each portion of the operation. The RMMV conducts mine detection and classification, after which the MH-60S conducts mine reacquisition, identification, and neutralization. For the purposes of the simulation,

the RMMV operates similarly to the MH-53E in the first stage of MCM-1 configuration simulations, beginning at the bottom left of the minefield and proceeding to the right, conducting mine detection and classification for each potential mine. Once the system has reached its maximum sortie time, it transits back to the staging area. This prompts the PMA sequence, which creates a list of targets for the MH-60S. The MH-60S then operates similarly to the MH-53E in the second stage of the MCM-1 configuration simulations, proceeding to each target as prescribed by a nearest neighbor algorithm and conducting mine reacquisition, identification, and neutralization. The LCS simulation uses the same general set of variables as the MCM-1 simulations (although fewer total variables are required because the LCS simulations do not require values for airborne mine detection or classification or values for surface mine reacquisition, identification, or classification). Figure 67 presents a visual representation of the simulation model for the LCS configurations, where the MH-60S is conducting mine neutralization activities in an area previously searched by the RMMV.

Figure 67 Detection-Neutralization Sequence: LCS Configurations

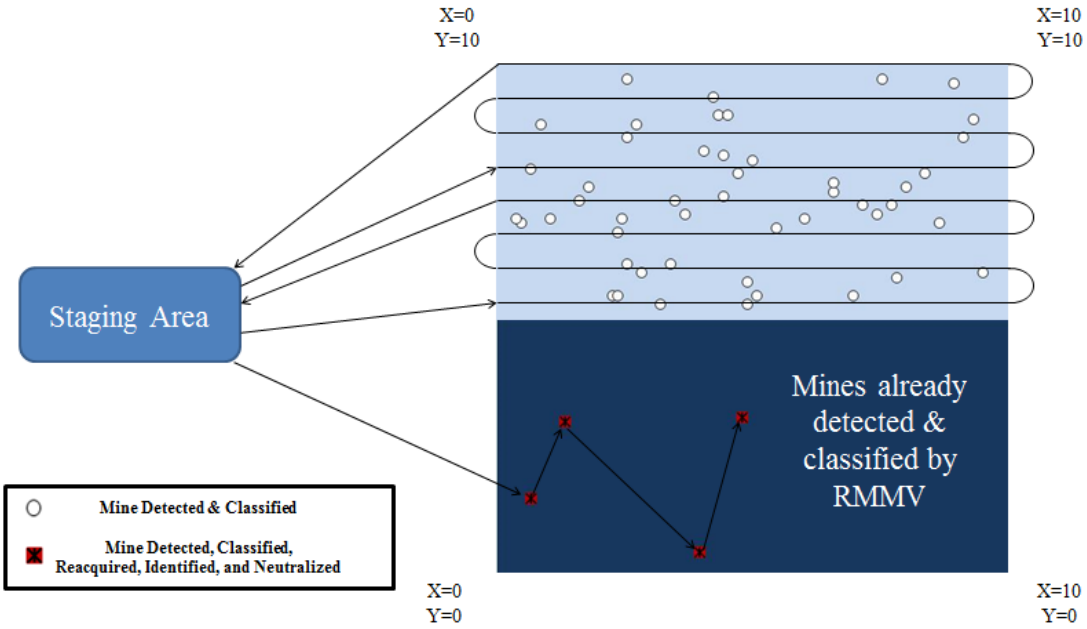


Table 8 and Table 9 present a full list of variable names, maximum values, and minimum values (discrete variables are denoted using asterisks). To aid organization, the functions and activities are grouped into three broad categories as defined previously in the MBSE MEASA: Design Variables, Operational Variables, and Environmental Variables. Note that for presentation the set of design variables has also been segmented into surface and airborne design variables. The variables are defined for each model: MCM-1 Avenger Configurations and LCS Configurations.

Table 8 Input Variable Summary: MCM-1 Configurations

Surface Asset Design Variables		
Variable	**Minimum**	**Maximum**
Transit to Area Speed (kts)	5	10
Transit to Minefield Speed (kts)	5	15
Search Gear Stream Time (hrs)	0.25	2
Search Gear Recovery Time (hrs)	0.25	2
Turn Time (seconds)	300	600
Maintenance Time (hours)	24	48
Sortie Time (hours)	350	500
RI&N Deployment Time (hours)	0.1	2
RI&N Recovery Time (hours)	0.1	2
Search Speed (kts)	2.5	5
Number of Tracks (per NM)*	20	25
Probability of Detection	0.7	0.9
Probability of Classification (Mine as MILCO)	0.7	0.9
Probability of Classification (Non-Mine as Non-MILCO)	0.7	0.9
Probability of Reacquisition (MILCO)	0.7	0.9
Probability of Reacquisition (Non-MILCO)	0.1	0.2
Probability of Identification (MILCO)	0.7	1
Probability of Identification (Non-MILCO)	0.7	1
Probability of Neutralization	0.7	1
Reaquisition and Identification Mean Time (hours)	0.25	1
Reaquisition and Identification Standard Deviation (hours)	0.1	0.5
Neutralization Mean Time (hours)	0.25	1
Neutralization Standard Deviation (hours)	0.1	0.5
Neutralizer Speed (kts)	2.5	5

Airborne Asset Design Variables		
Variable	**Minimum**	**Maximum**
Transit to Area Speed (kts)	N/A	N/A
Transit to Minefield Speed (kts)	90	180
Search Gear Stream Time (hrs)	0.2	0.5
Search Gear Recovery Time (hrs)	0.2	0.5
Turn Time (seconds)	120	240
Maintenance Time (hours)	1	2
Sortie Time (hours)	3	5
RI&N Deployment Time (hours)	0.25	0.5
RI&N Recovery Time (hours)	0.25	0.5
Search Speed (kts)	15	30
Number of Tracks (per NM)*	20	25
Probability of Detection	0.7	0.9
Probability of Classification (Mine as MILCO)	0.7	0.9
Probability of Classification (Non-Mine as Non-MILCO)	0.7	0.9
Probability of Reacquisition (MILCO)	0.7	0.9
Probability of Reacquisition (Non-MILCO)	0.1	0.2
Probability of Identification (MILCO)	0.7	1
Probability of Identification (Non-MILCO)	0.7	1
Probability of Neutralization	0.7	1
Reaquisition and Identification Mean Time (hours)	0.25	0.5
Reaquisition and Identification Standard Deviation (hours)	0.1	0.25
Neutralization Mean Time (hours)	N/A	N/A
Neutralization Standard Deviation (hours)	N/A	N/A
Neutralizer Speed (kts)	2.5	5

Operational Variables		
Variable	**Minimum**	**Maximum**
Number of Minefield Passes*	1	3
Surface Search Percentage	0.3	0.7
Surface Neutralization Percentage	0.3	0.7

Environmental Variables		
Variable	**Minimum**	**Maximum**
Transit Distance (miles)	100	500
Staging Area Distance (miles)	10	20

Table 9 Input Variable Summary: LCS Configurations

Surface Asset Design Variables		
Variable	**Minimum**	**Maximum**
Transit to Area Speed (kts)	15	40
Transit to Minefield Speed (kts)	10	40
Search Gear Stream Time (hrs)	0.25	2
Search Gear Recovery Time (hrs)	0.25	2
Turn Time (seconds)	300	600
Maintenance Time (hours)	2	4
Sortie Time (hours)	10	20
RI&N Deployment Time (hours)	N/A	N/A
RI&N Recovery Time (hours)	N/A	N/A
Search Speed (kts)	5	15
Number of Tracks (per NM)*	20	25
Probability of Detection	0.7	0.9
Probability of Classification (Mine as MILCO)	0.7	0.9
Probability of Classification (Non-Mine as Non-MILCO)	0.7	0.9
Probability of Reacquisition (MILCO)	N/A	N/A
Probability of Reacquisition (Non-MILCO)	N/A	N/A
Probability of Identification (MILCO)	N/A	N/A
Probability of Identification (Non-MILCO)	N/A	N/A
Probability of Neutralization	N/A	N/A
Reaquisition and Identification Mean Time (hours)	N/A	N/A
Reaquisition and Identification Standard Deviation (hours)	N/A	N/A
Neutralization Mean Time (hours)	N/A	N/A
Neutralization Standard Deviation (hours)	N/A	N/A
Neutralizer Speed (kts)	N/A	N/A

Airborne Asset Design Variables		
Variable	**Minimum**	**Maximum**
Transit to Area Speed (kts)	N/A	N/A
Transit to Minefield Speed (kts)	90	180
Search Gear Stream Time (hrs)	N/A	N/A
Search Gear Recovery Time (hrs)	N/A	N/A
Turn Time (seconds)	120	240
Maintenance Time (hours)	1	2
Sortie Time (hours)	3	5
RI&N Deployment Time (hours)	0.25	0.5
RI&N Recovery Time (hours)	0.25	0.5
Search Speed (kts)	15	30
Number of Tracks (per NM)*	N/A	N/A
Probability of Detection	N/A	N/A
Probability of Classification (Mine as MILCO)	N/A	N/A
Probability of Classification (Non-Mine as Non-MILCO)	N/A	N/A
Probability of Reacquisition (MILCO)	0.7	0.9
Probability of Reacquisition (Non-MILCO)	0.1	0.2
Probability of Identification (MILCO)	0.7	1
Probability of Identification (Non-MILCO)	0.7	1
Probability of Neutralization	0.7	1
Reaquisition and Identification Mean Time (hours)	0.25	0.5
Reaquisition and Identification Standard Deviation (hours)	0.1	0.25
Neutralization Mean Time (hours)	0.25	1
Neutralization Standard Deviation (hours)	0.1	0.5
Neutralizer Speed (kts)	2.5	5

Operational Variables		
Variable	**Minimum**	**Maximum**
Number of Minefield Passes*	1	3
Surface Search Percentage	0.3	0.7
Surface Neutralization Percentage	0.3	0.7

Environmental Variables		
Variable	**Minimum**	**Maximum**
Transit Distance (miles)	100	500
Staging Area Distance (miles)	10	20

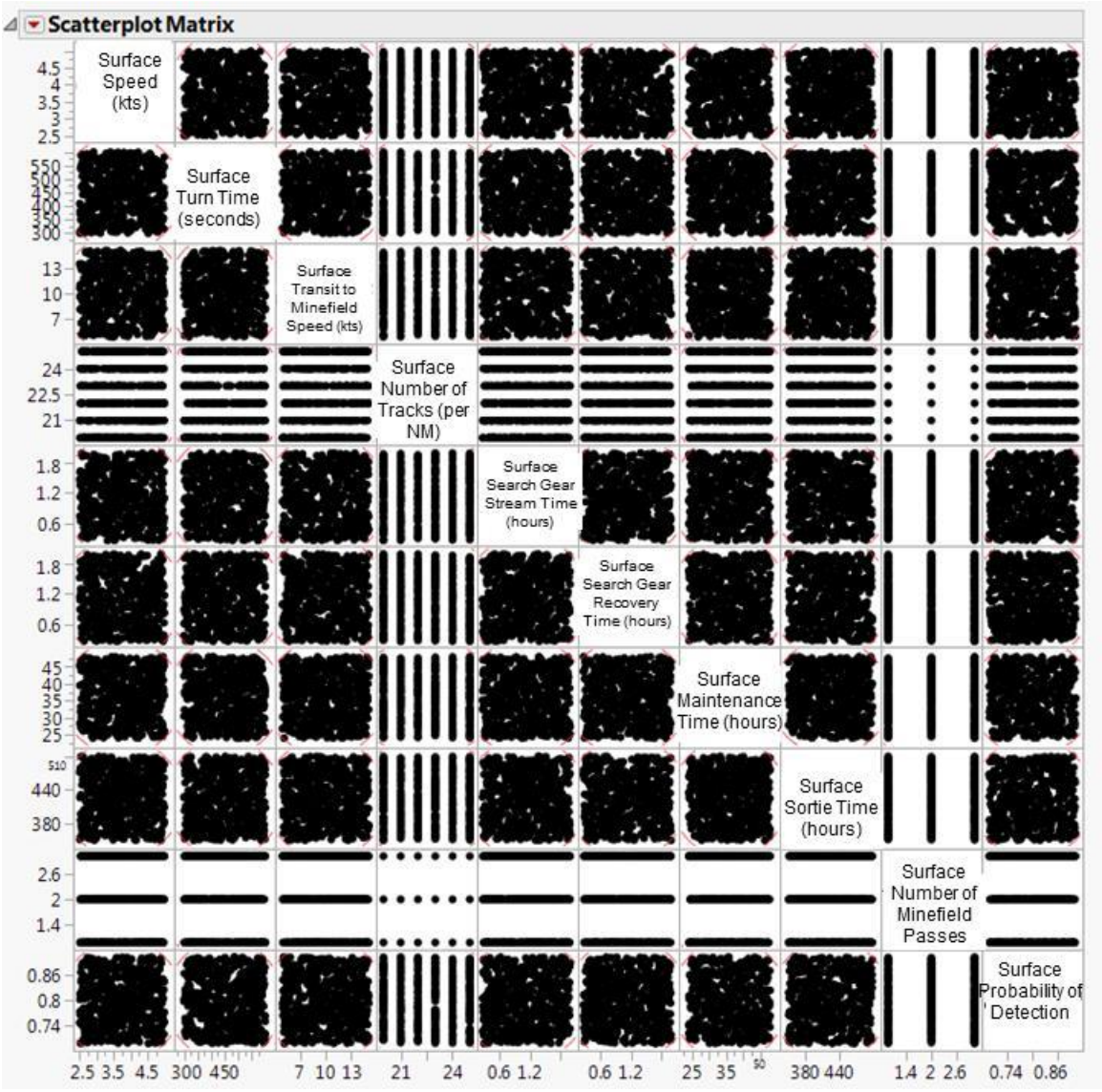
2. Experimental Design Selection

The Active, Defensive MCM operation discrete event simulation is characterized by 51 input variables for the MCM-1 Configurations and 32 input variables for the LCS configurations (as noted in the previous section, the difference arises from the operating procedures of each configuration, for example the LCS does not conduct airborne mine detection, therefore the variable for the probability of airborne mine detection is not needed for the LCS configurations). Per Figure 48 (presented in Chapter III), utilization of a 512 design point NOB design is well suited for this situation. These designs allow for both discrete and continuous variables, provide excellent space filling properties across the design space, and allow for maximum flexibility during model fitting after the simulation has been run.

Vieira et al. (2011) provides a summary of the importance of minimal correlation and minimum imbalance for a space filling design with both discrete and continuous factors. As a brief review, designs with correlations between factors introduce the possibility of mischaracterizing the relationship between input variables and output variables. Accordingly, designs with near zero correlation between columns of the experimental design matrix are preferred. Equally important when a simulation must consider both continuous and discrete variables is the balance between columns of the experimental design matrix. As presented in Vieira et al. (2011), when a design intended to be used solely for continuous factors is used for discrete factors, rounding of each design point is required. While some rounding may be acceptable, this rounding has the potential to increase the correlation between columns of the experimental design matrix. The design methodology presented in Vieira et al. (2011) defines a procedure for creating designs for both continuous and discrete factors and presents an imbalance criterion, where designs with near zero imbalance between columns of the experimental design matrix are preferred. A scatterplot matrix for the ten surface search variables for the MCM-1 configurations variables is presented in Figure 68 to provide visual confirmation that the design provides adequate space filling between variables, the correlation and space filling are subsequently assessed numerically. Notice that eight of the variables are continuous; therefore nearly all of the space is filled with design points. Two variables

(Number of Tracks per Nautical Mile and Number of Minefield Passes) are discrete variables, and are therefore only tested at six and three levels, respectively. Examination of the full correlation matrix showed a maximum absolute pairwise correlation of 0.0266 and a maximum imbalance of 0.1015, suggesting that correlation between input variables and imbalanced testing of discrete variables is not an issue for this design.

Figure 68 Scatterplot Matrix (First Ten Simulation Variables)



Each of the 512 design points prescribed by the NOB designs was replicated 30 times for each model. This resulted in a total of 15,360 runs for each model and an overall total of 30,720 model runs. This replication of design points is important for stochastic simulations, and allows for an examination of the variability at each design point within each model (if that is also of interest).

C. MODEL ANALYSIS

The second goal of this dissertation research is to demonstrate the utility of the MBSE MEASA through an analysis of a U.S. Navy system. In particular, this research analyzes the operational effectiveness of the MCM-1 Avenger MCM System and the LCS MCM System. Per the MBSE MEASA, the goal of that analysis is to establish a relationship between system design parameters (as well as operational and environmental factors) and operational MOEs.

1. Effectiveness Definition

As presented earlier in this chapter, the MCM simulation model assesses the ability of different MCM configurations to complete an Active, Defensive MCM operation. Accordingly, measures of effectiveness are required that quantify the mission accomplishment capabilities of the system in this environment. Detailed review of mine warfare guidance, in particular NWP 3–15, suggests that traditional MCM metrics focus on the idea of "residual risk," which is informally defined as the probability that something remains in the minefield. This naturally leads to the first measure of effectiveness used in this analysis, specifically the percentage of mines cleared. This is also the basis of the traditionally used mine countermeasures metric, the area coverage rate sustained (ACRS), which is defined as the ratio of the area covered during an operation and the operational duration. Becker et al. (2014) utilize these metrics and demonstrate that the probabilities of detection, classification, identification, and neutralization dominate performance in terms of percent clearance and ACRS. However, while these metrics capture the idea of "residual risk" quite well, the broad range of values assigned to the probabilities of detection, classification, identification, and neutralization may result in a compounding of error if the probabilities modeled do not

correspond to the true performance of the systems of interest (recall that utilizing a broad range is still considered preferable to using actual performance data, which would result in classification of the results). In order to conduct analysis that reduces the impact of the potential compounding of error a new metric is introduced, specifically the probability that the system achieves 90% detection of mines in the minefield. This metric captures the idea of residual risk (the probability that a mine is undetected) and also highlights the potential impact of multiple passes through the minefield. Furthermore, while the potential issue of an incorrect specification of detection probability remains; incorrect specification of classification, identification, and neutralization probabilities does not compound the issue.

1. Percent Clearance
2. Area Coverage Rate Sustained
3. Probability of 90% Detection

Analysis of these three measures of effectiveness aligns with the guidance specified in standard mine warfare guidance. Use of these three measures also tailors the analysis to capture the full range of behaviors specified in the architecture products and also acknowledges the potential limitations of the simulation model resulting from the choice not to input classified system design parameter data.

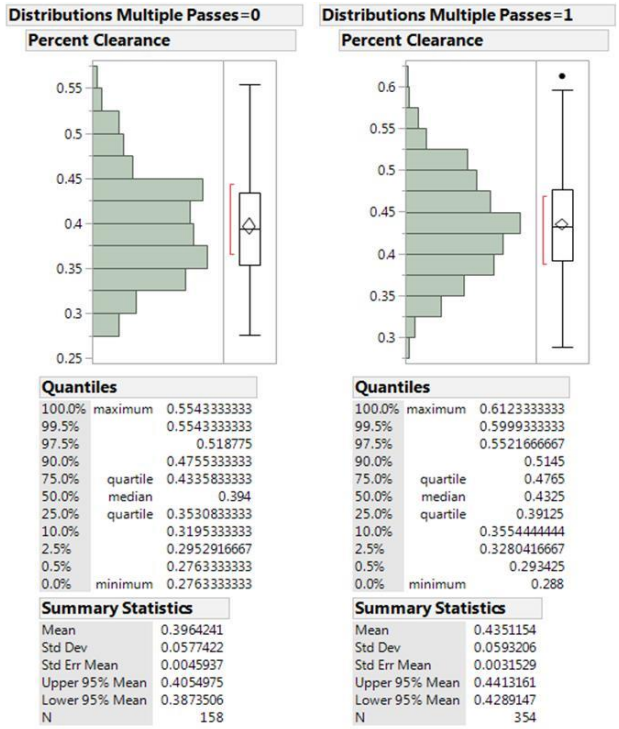
a. MCM-1 Model Analysis

Analysis of the MCM-1 model determines the input variables that have the greatest impact on each of the three output variables presented earlier. Initial analysis of the Percent Clearance metric show similar results to the previous study, which suggest that the probabilities of detection, classification, identification, and neutralization have a substantial impact on the percentage of mines cleared. Regression analysis (Figure 99, Figure 100, and Figure 101 in Appendix E) shows that the number of passes through the minefield (an operational variable) has a substantial impact on each of the Operational MOEs.

This analysis also explores the impact of multiple passes through the minefield to better characterize the variables that have the largest impact on system performance. Given that the regression analysis suggests that there is little difference between

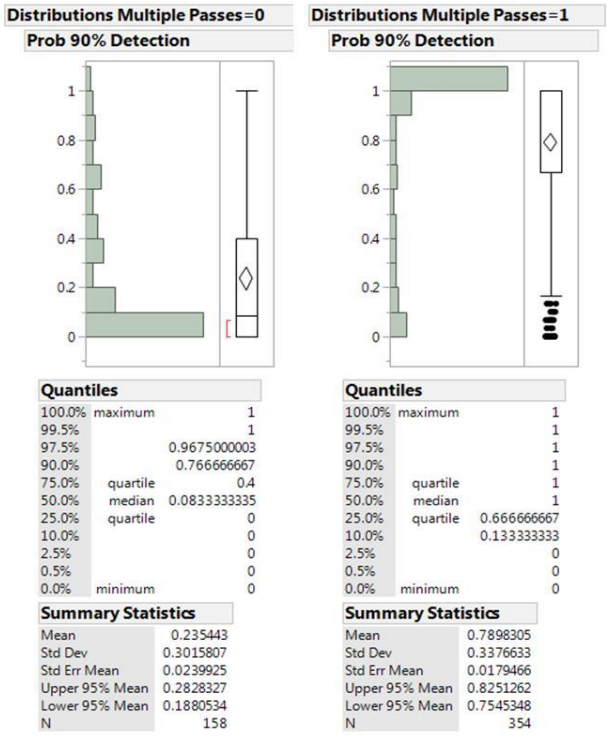
conducting two or three passes through the minefield, these simulation runs are grouped and compared to the simulation runs conducting only one pass through the minefield. Figure 69 presents two histograms that reinforce the difference between configurations conducting multiple minefield passes and configurations conducting only a single minefield pass. It also demonstrates that the approximately 4% increase in Percent Clearance can be seen across the interquartile range and at the maximum Percent Clearance values.

Figure 69 Histogram Comparison of Percent Clearance for Single versus Multiple Minefield Passes (MCM-1 Configurations)



While multiple minefield passes demonstrate some potential value in terms of percent clearance, multiple passes are likely associated with some increase to operational duration and cost. Examination of histograms comparing the performance for each potential configuration in terms of the second metric of interest, the Probability of 90% Detection, suggests a more distinct difference between the configurations (Figure 70).

Figure 70 Histogram Comparison of Probability of 90% Detection for Single versus Multiple Minefield Passes (MCM-1 Configurations)



Initial data examination suggests that the selection of Probability of 90% Detection is a potentially illuminating measure of effectiveness beyond the Percent Clearance for this simulation. Examination of Figure 70 shows that configurations conducting multiple minefield passes increase the Probability of 90% Detection by 55%. Perhaps more importantly, the median Probability of 90% Detection for configurations conducting multiple passes is 100%.

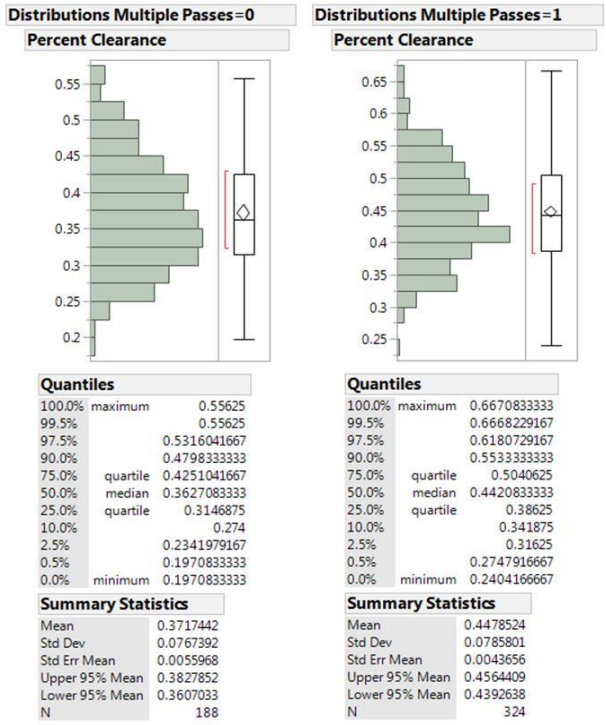
While the realization that multiple passes through the minefield improves system performance is not shocking, it is useful for two reasons. First, from a practical perspective, the stark difference between scenarios where only one minefield pass is conducted and scenarios where multiple passes are conducted may be a useful demonstrator of the importance of allowing time to conduct Active, Defensive MCM Operations. Second, the objective of the Model Analysis step of the MBSE MEASA is to identify feasible system configurations. The breadth of the operational simulations advocated by the MBSE MEASA, which consider not only system design parameters, but

also operational and environmental factors, can complicate this identification. This can present challenges when presenting results because it is possible to consider these factors in either order. It is possible to hold operational and environmental factors constant and examine the impact of changes to the system design parameters to identify preferred system configurations for a specific implementation, and it is also possible to hold the system configurations constant and examine the impact of changes to system operational implementation to identified preferred methods of operation given a constant system. A third approach, called robust design, allows environmental factors and other "noise" factors to vary, and examines the expected mean and variability of performance across these noise conditions. Subsequently, the analyst can seek to identify system design parameters that yield solutions with robust performance. This dissertation does not use a robust design approach, but the approach, developed by noted engineer and statistician Genichi Taguchi in the 1960s, is frequently applied in industrial applications for product design. A description of the utility of robust design approaches is presented in Sanchez (2000) and an application of robust design for multi-nation mine clearing is presented in Thompson (2015). In this case conducting a single pass through the minefield results in a near zero probability of achieving the desired level of system performance (in terms of mine detection), therefore future analysis focuses on identification of feasible system configurations that conduct multiple minefield passes. A similar analysis is conducted for the LCS MCM configurations to facilitate development of tradespace visualization tools per Step 5 of the MBSE MEASA.

b. LCS Model Analysis

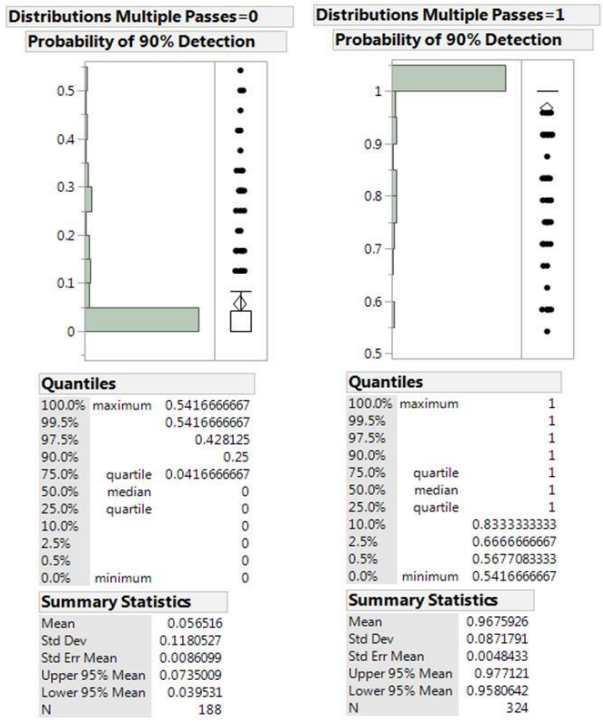
As with the MCM-1 model data, examination of the LCS model data determines the variables that have the greatest impact on each operational MOE. Regression analysis results (Presented in Figure 102, Figure 103, Figure 104, and Figure 105 in Appendix E) suggest that, as with the MCM-1 configurations, it may be interesting to examine the impact of conducting multiple passes through the minefield. Histogram analysis (Figure 71) suggests that there is a 7% improvement to Percent Clearance resulting from multiple passes both at the median clearance level as well as across the full range of Percent Clearance.

Figure 71 Histogram Comparison of Percent Clearance for Single versus Multiple Minefield Passes (LCS Configurations)



As with the MCM-1 configurations, detailed analysis of the distribution of the second operational MOE (the Probability of 90% Detection) provides additional insight. While there are only minimal differences in terms of the percent clearance, there is a substantial difference in distributions for the Probability of 90% Detection for configurations that conduct a single minefield pass and configurations that conduct multiple passes (Figure 72).

Figure 72 Histogram Comparison of Probability of 90% Detection for Single versus Multiple Minefield Passes (LCS Configurations)



For the LCS configurations conducting multiple minefield passes, the average Probability of 90% Detection jumps from less than 6% to over 95% compared to configurations conducting only a single minefield pass. Furthermore, the minimum Probability of 90% Detection conducting multiple minefield passes is actually equal to the maximum Probability of 90% Detection when conducting only a single minefield pass. This emphasizes the impact that operational decisions can have on system performance, even in terms of simulation models. While it is certainly not groundbreaking that searching a minefield more thoroughly results in improvements to system performance, this analysis demonstrated that a simple alteration to an operational factor had a substantial impact on operational performance, dominating the impact of alterations to system design parameters. Furthermore, this analysis suggests that when comparing potential changes to system design parameters such as the probability of identification or the probability of neutralization using a tradespace visualization tool, it is prudent to include the operational factor of the number of minefield passes in the

visualization tool to highlight the set of feasible system design parameter configurations subject to operational decisions.

2. Tradespace Analysis

The final step of the MBSE MEASA suggests development of a tradespace visualization tool to allow for examination of system tradeoff decisions from multiple perspectives. In particular, the methodology advocates development of external operational and system synthesis models to allow for definition of a set of system configurations that are feasible from an operational, physical, and cost perspective. In this particular case, development of physical models are not necessary, given that the analysis is focused on a comparison of systems that exist and alterations to their physical design are unrealistic. However, the cost modeling and analysis of the MCM-1 Avenger and the LCS presented in Becker at al. (2014) can be used in conjunction with the operational effectiveness analysis modeling and analysis presented in this research to develop a tradespace visualization tool that highlights a set of feasible system configurations in terms of both operational effectiveness and operational cost (a function of operational duration). While the example tool presented in Chapter III assumes that a single system is being developed (and therefore a single tradespace visualization tool is sufficient), this demonstration considers two distinct systems, therefore two distinct tradespace visualization tools are required (one for the MCM-1 configurations and one for the LCS configurations).

Figure 73 shows an operational tradespace visualization approach for defining a feasible set of system configurations for the MCM-1 Avenger, focused solely on operational MOEs. Note that the prediction formulas developed in the regression analysis shown in Appendix E for each of the operational MOEs are used as surrogate models to facilitate rapid updating of the tool. Recall that the visualization approach presents two-dimensional projections of the larger, multi-dimensional tradespace. Figure 73 assumes that a threshold of 90% has been established for the Probability of 90% Detection, a threshold of 0.20 has been established for the ACRS, and threshold of $15 million has

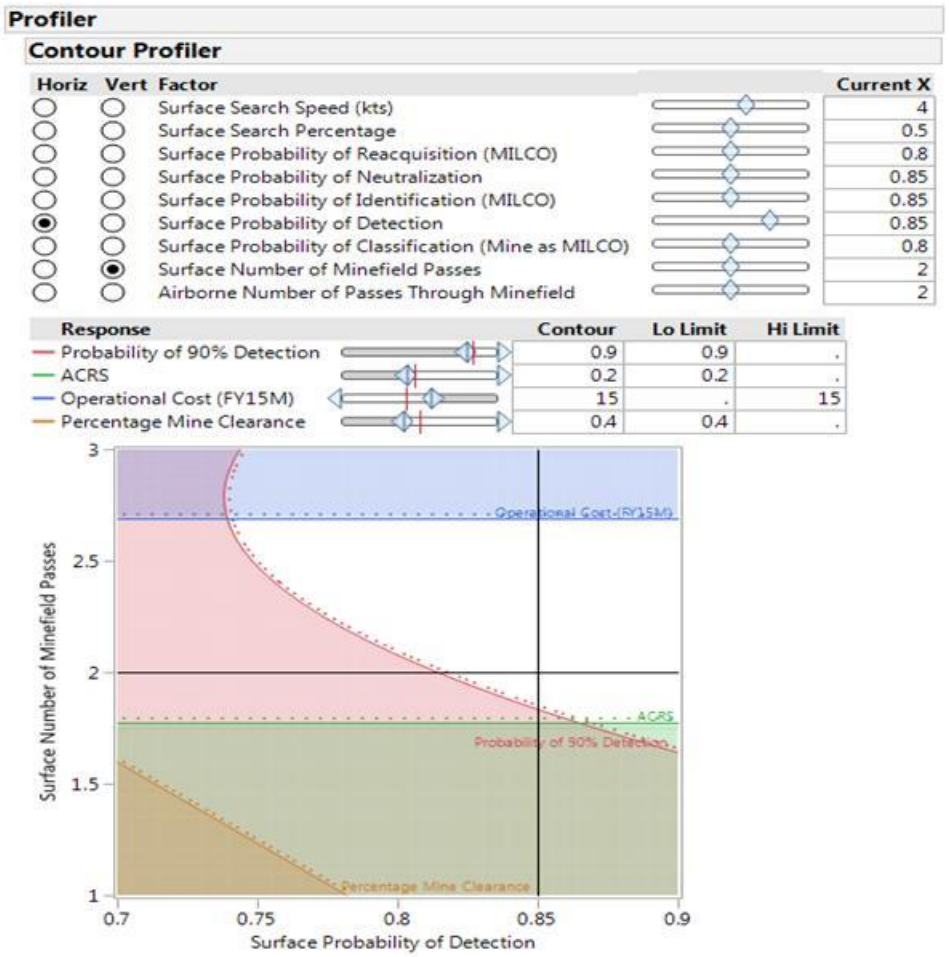
been established for the Operational cost, and a threshold of 40% has been established for the Percent Clearance.

Note that the system design parameters, environmental factors, and operational factors identified as having a significant effect on the operational effectiveness of the MCM-1 Avenger for each of the operational MOEs shown earlier are shown as "Factors" in Figure 73. Selection of which factor is shown on the Horizontal and Vertical axis is accomplished by interaction with the selection bubble next on the left side of each factor. Each factor not shown on either the x-axis or y-axis is held constant at the value shown to the right of each factor name (note that each factor is initially set at the mean of the minimum and maximum values shown in Table 8 and Table 9). The current settings define a feasible region (shown in white) in terms of the Probability of Detection (x-axis) and the Number of Minefield Passes (y-axis) for MCM-1 configurations, assuming that each of the other factors is fixed at the value shown.

The importance of setting each of these factors at a constant value cannot be overemphasized. As mentioned, there are millions of potential combinations of factors that may be investigated. This research presents tradespace visualization and investigation as an alternative for system design to emphasize that the goal of analysis at this stage should be to reduce the potential tradespace by identifying factor combinations that are infeasible, rather than driving toward a specific system configuration. Additional research is required to investigate efficient techniques for the investigation of large, multi-dimensional tradespaces. As mentioned previously, Ross (2003) defines a multi attributed trade space exploration procedure that documented a method for the definition of a Pareto frontier of solutions. That work was expanded to a 48 step multi-attribute trade space exploration process and demonstrated more recently in Ross, Stein, and Hastings (2014) and applied to survivability analysis of satellite systems. The process is intended to be implemented for communication with stakeholders and accordingly only utilizes more traditional factorial designs to conduct detailed modeling and simulation. Integration of that approach, which demonstrates that it is possible to quickly reduce the size of a system tradespace through interaction with stakeholders as well as simulation modeling, with the MBSE MEASA developed in this research is an area of potentially

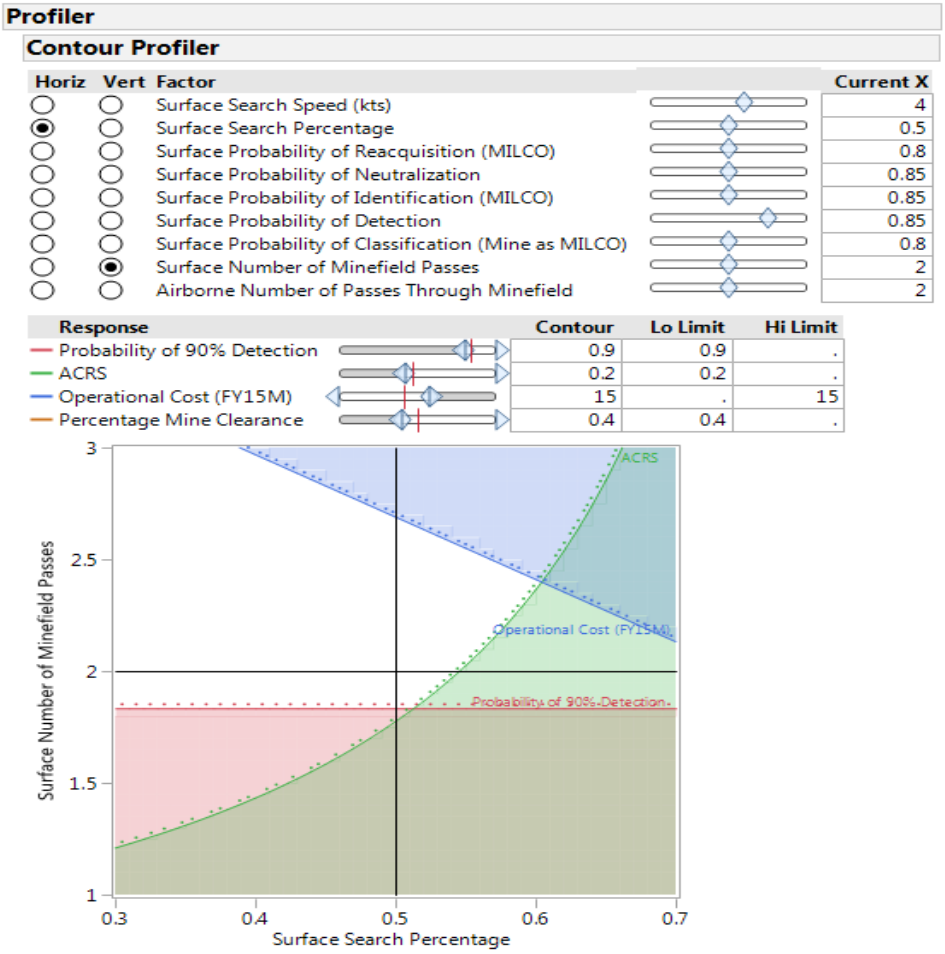
interesting future research. For the purposes of this dissertation, a simple demonstration of the utility of tradespace exploration is presented that focuses primarily on several of the most significant performance drivers (as identified by the regression analysis), the probability of detection, the number of minefield passes, the search speed, and (for the MCM-1) the Surface Search Percentage as well as (for the LCS) the Surface Sortie Time. Note that each of the other factors are held constant in this example (as mentioned, the Probabilities of Classification, Reacquisition, Identification, and Neutralization are held constant at the mean of the ranges presented in Table 8and Table 9) and the conclusions identified in this approach are only valid given the fixed values of each of those factors.

Figure 73 Operational Tradespace Visualization (View 1): MCM-1 Configurations



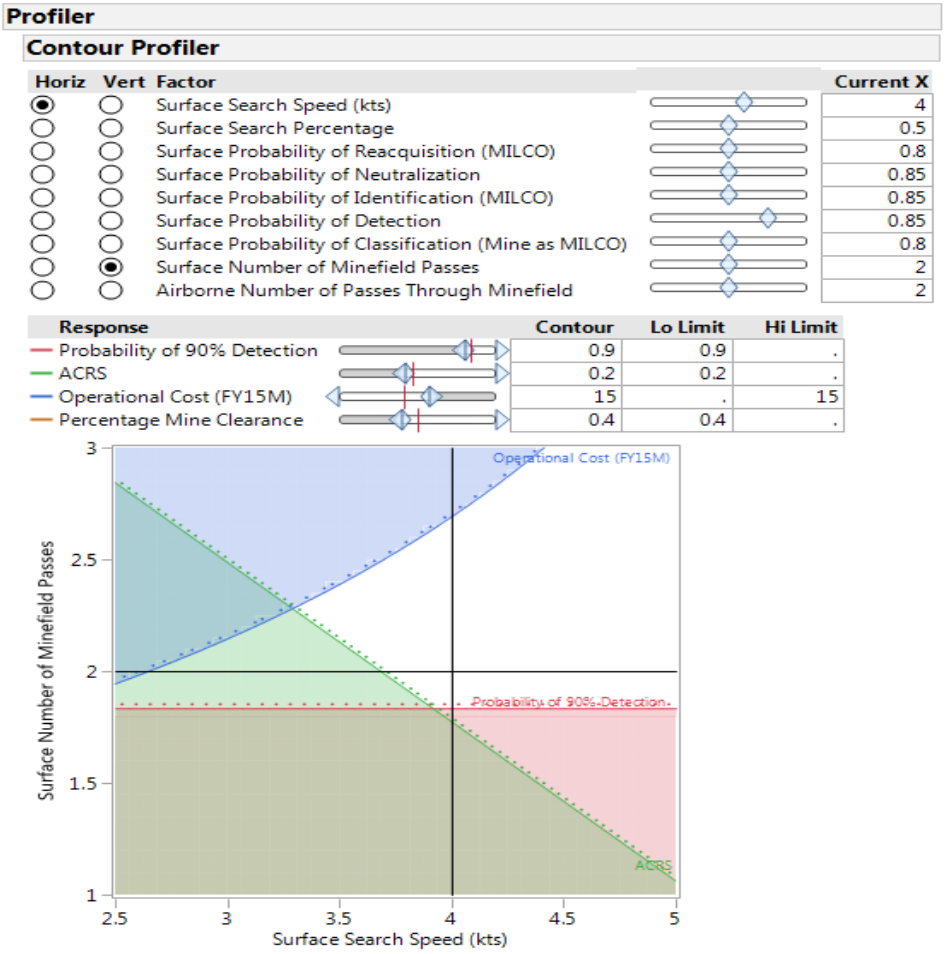
Note that the system MOE for cost is presented in the same window as the operational MOEs to conserve space. There are several major conclusions that can be drawn from examination of Figure 73. First, the system requires a Probability of Detection of at least approximately 0.83 to ensure that the threshold of 90% Probability of 90% Detection is met. Second, two minefield passes must be conducted to satisfy the thresholds for Probability of 90% Detection and ACRS. Notably, three minefield passes cannot be conducted due to the threshold imposed for the Operational Cost. Recall that this tradespace for Probability of Detection and Number of Minefield Passes exists given the values set for each of the other factors in Figure 73. This analysis follows the exploration approach outlined in Chapter III, and once as many two-dimensional projections as possible are explored could be used to define a set of feasible system design parameters. As a point of caution, note that there is variability associated with the prediction formulas used to generate the tradespace shown in Figure 73 and accordingly it is imprudent to make specific recommendations at the constraint boundaries (the same is true for subsequent tradespace visualizations). The goal of examining these tradespaces should be to identify portions of the tradespace that are infeasible, rather than to recommend a particular system configuration. This should facilitate development of more refined system requirements (ex: Probability of Detection greater than 0.80) that can then be used to bound future iterations of system analysis. It is possible to examine alternate two-dimensional projections that may change the conclusions drawn from Figure 73. Figure 74 presents a visualization of the tradespace between the Surface Search Percentage (x-axis) and the Number of Minefield Passes (again on the y-axis).

Figure 74 Operational Tradespace Visualization (View 2): MCM-1 Configurations



Notice that the shape of the tradespace is completely different from Figure 73 as a result of the change to the two-dimensional projection being examined. Once again, a single minefield pass is incapable of satisfying the threshold for the Probability of 90% Detection or ACRS. However, a third minefield pass is now possible if the Surface Search Percentage is reduced below approximately 0.40. This suggests that the Operational Cost is dependent on the operational decision to have the surface asset search a larger portion of the minefield but not on the Probability of Detection, which makes intuitive sense. It is possible to examine an additional two-dimensional projection (Figure 75) that again shows the Number of Minefield Passes on the y-axis and now shows the Surface Search Speed on the x-axis.

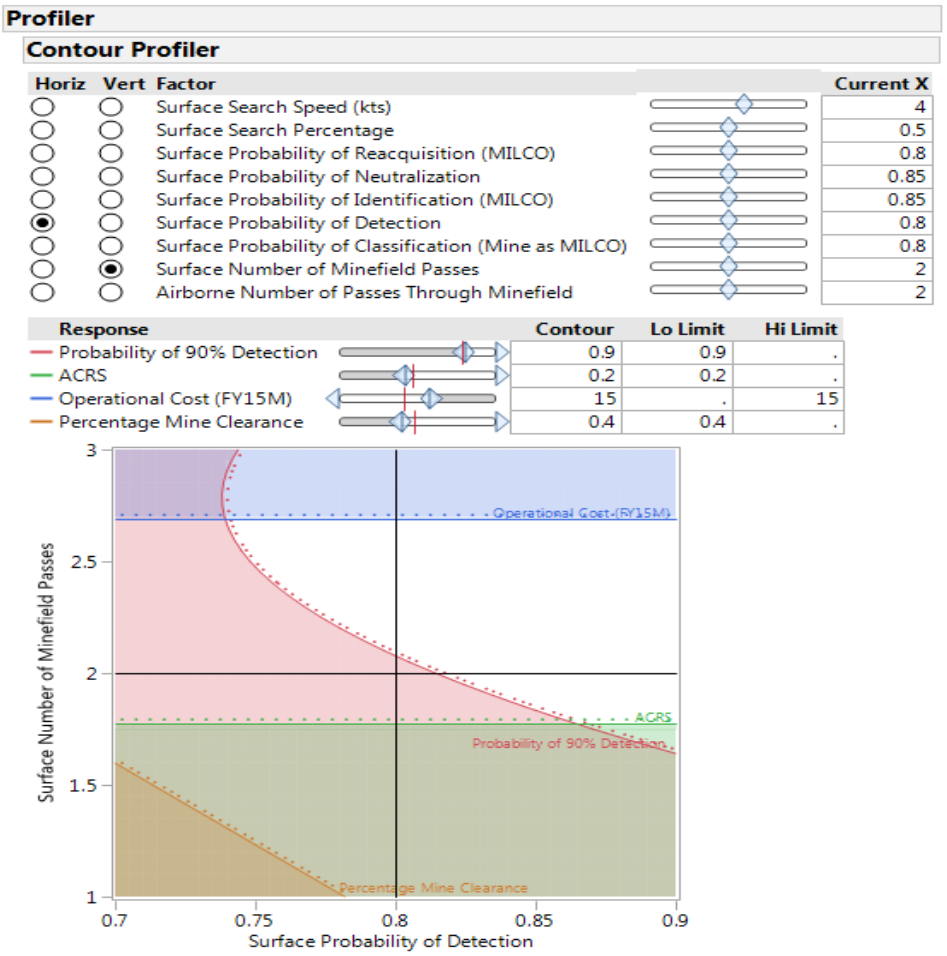
Figure 75 Operational Tradespace Visualization (View 3): MCM-1 Configurations



Notice that the Probability of 90% Detection threshold and the ACRS threshold both once again suggest that a single minefield pass in infeasible. If two minefield passes are conducted a Search Speed of approximately 3.5 knots is required. Knowledge of these alternative two-dimensional projections is helpful when the original two-dimensional project (showing the Number of Minefield Passes and the Probability of Detection) is reexamined and modified. Recall that Figure 73 suggested that, for the given values of each factor, a Probability of Detection of approximately 0.83 is required. However, it is useful to demonstrate how the tradespace visualization tool can aid decision making when the configurations initially defined as feasible cannot actually be realized. For example, there may be a scenario where the Probability of Detection is restricted to 0.80.

Figure 76 presents a visualization of the original two-dimensional projection with the Probability of Detection restricted to 0.80.

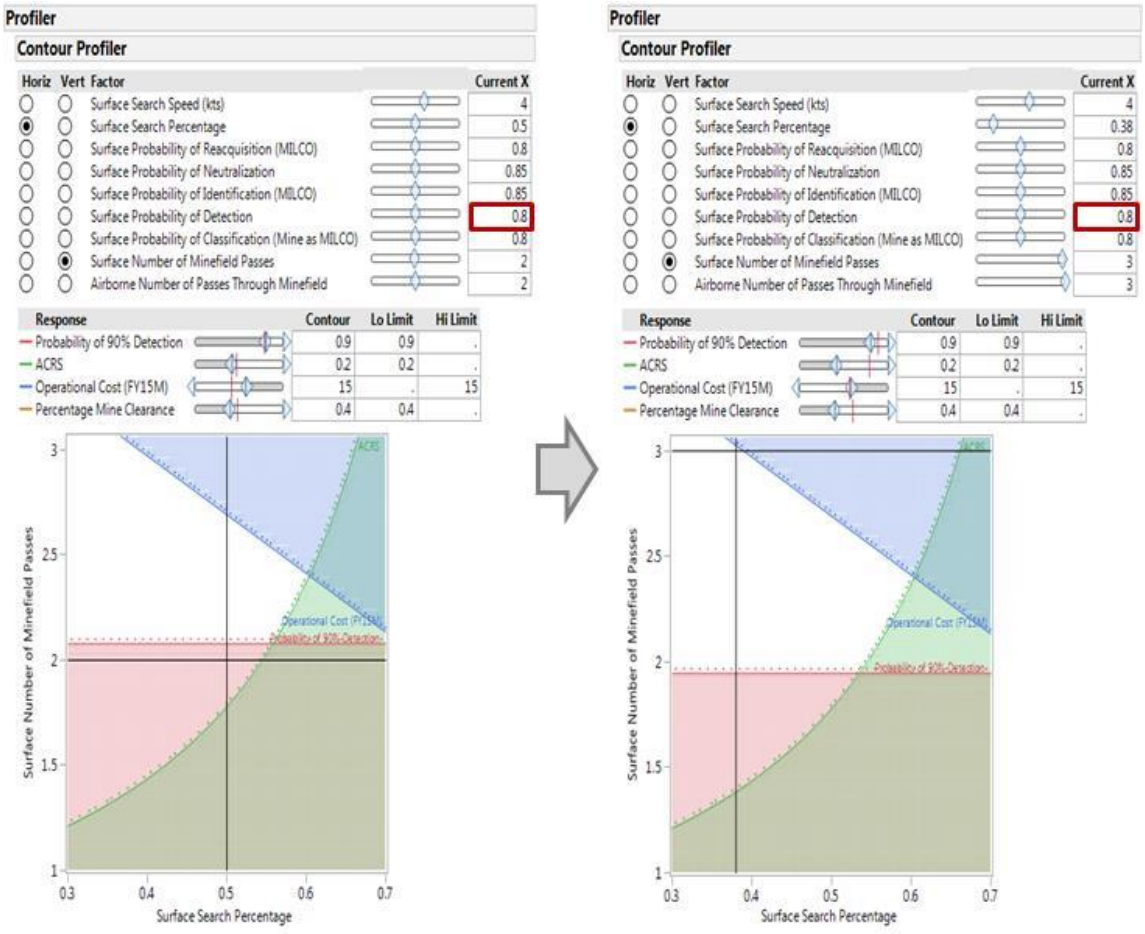
Figure 76 Operational Tradespace Visualization (View 4): MCM-1 Configurations



Notice that the crosshair now suggests that the system is incapable of satisfying the Probability of 90% Detection threshold with a Probability of Detection of 0.80 (note that an additional minefield pass is also infeasible due to the Operational Cost constraint. It is possible to increase the size of the feasible region by relaxing one or more constraints it is also possible to increase the size of the feasible region by altering the settings for factors other than the Number of Minefield Passes and the Probability of Detection. Figure 77 presents two screenshots of the two-dimensional projection original presented in Figure 74

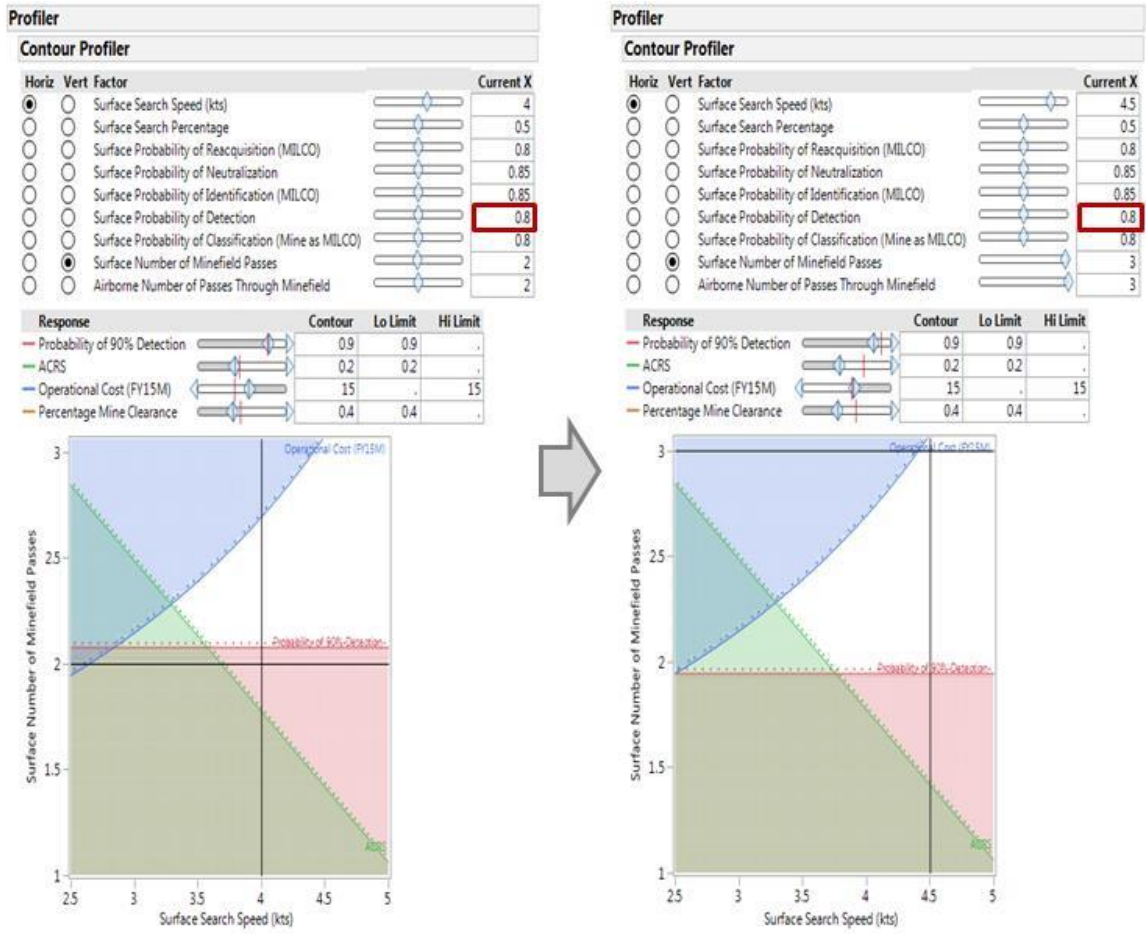
(where the Number of Minefield passes is shown on the y-axis and the Surface Search Percentage is shown on the x-axis). The left side of Figure 77 shows the resulting two-dimensional projection when the Probability of Detection is restricted to 0.80. Notice that the system is classified as infeasible for the Probability of 90% Detection MOE. However, when examining the tradespace, a third minefield pass can actually be conducted (something that was not apparent when examining the projection for the Number of Minefield Passes and the Probability of Detection) by decreasing the Surface Search Percentage. The right side of Figure 77 presents a visualization of such a solution, where a third minefield pass is conducted and the system is not infeasible for Operational Cost because the Surface Search Percentage has been reduced to 0.38 (previously it was 0.50).

Figure 77 Operational Tradespace Visualization (View 5): MCM-1 Configurations



While the reduction of Surface Search Percentage is certainly a legitimate solution when the Probability of Detection is restricted to 0.80 it is certainly not the only potential solution. Figure 78 presents a similar examination of the two-dimensional tradespace originally presented in Figure 75 (where the Number of Minefield Passes is shown on the y-axis and the Surface Search Speed is shown on the x-axis). On the left side of Figure 78 the system is infeasible due to the inability to meet the Probability of 90% Detection threshold due to the reduction of Probability of Detection (note that this example assumes that the Surface Search Percentage has been reset to 0.50). On the right side of Figure 78 a potential solution is identified, showing that the system can conduct a third minefield pass, which will satisfy the Probability of 90% Detection threshold, without exceeding the Operational Cost threshold by increasing the Surface Search Speed from 4 knots to 4.5 knots. Note that there are numerous potential two-dimensional projections that may be explored. This particular example focused on operational decisions regarding the Number of Minefield Passes, the Surface Search Percentage, and the Surface Search Speed that can be made to overcome a restriction on the Probability of Detection (assuming constant values for the Probabilities of Classification, Reacquisition, Identification, and Neutralization). Each user must make a decision regarding the appropriate ordering of factor investigation; however this example demonstrated the utility that a tradespace visualization approach can have for multi-attribute tradespaces.

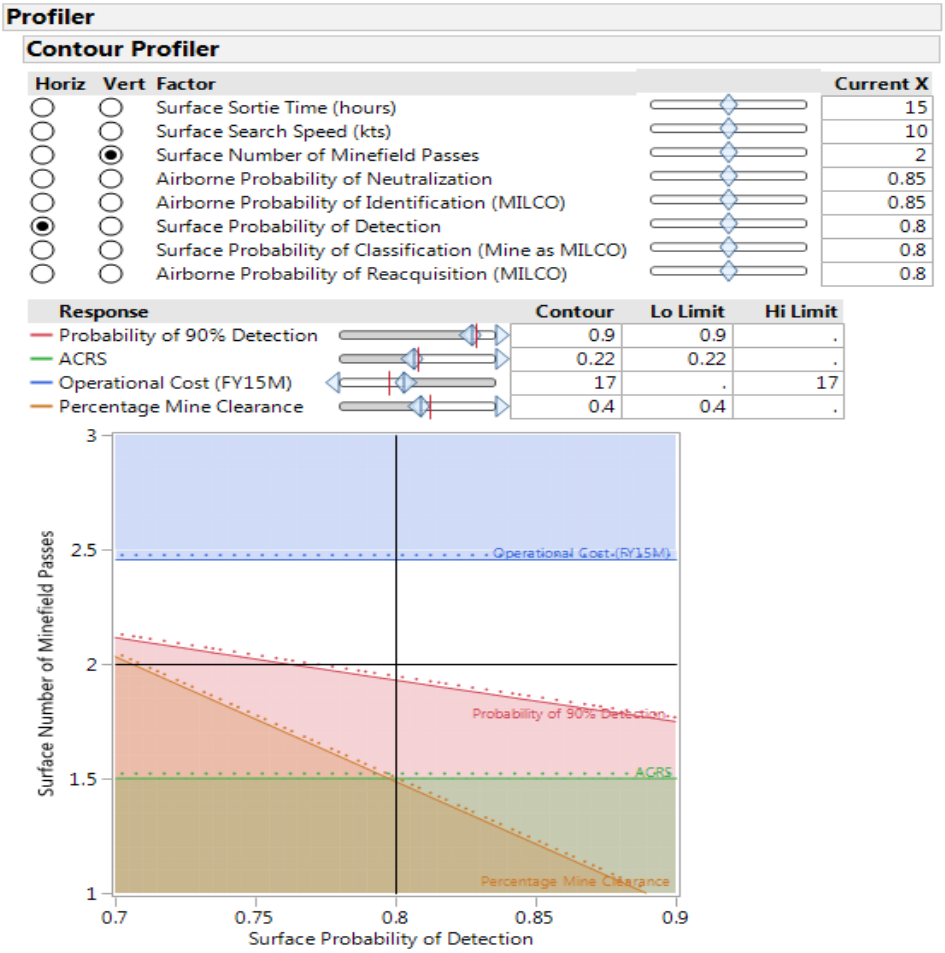
Figure 78 Operational Tradespace Visualization (View 6): MCM-1 Configurations



A similar analysis can be conducted for the LCS MCM configurations. Figure 79 presents a similar tradespace visualization approach for defining a set of feasible design parameters for LCS configurations. Once again, a threshold of 90% has been established for the Probability of 90% Detection and a 40% threshold has been established for the Percent Clearance. For the purposes of presenting an interesting demonstration, slightly altered thresholds were imposed for the ACRS and the Operational Cost. A threshold of 0.22 has been established for the ACRS (the LCS configurations demonstrated slightly better performance), and threshold of $17 million has been established for the Operational Cost (the LCS configurations demonstrated a reduced operational cost). Once again the demonstration

begins with the presentation of the two-dimensional tradespace (Figure 79) for the Number of Minefield Passes (y-axis) and the Probability of Detection (x-axis).

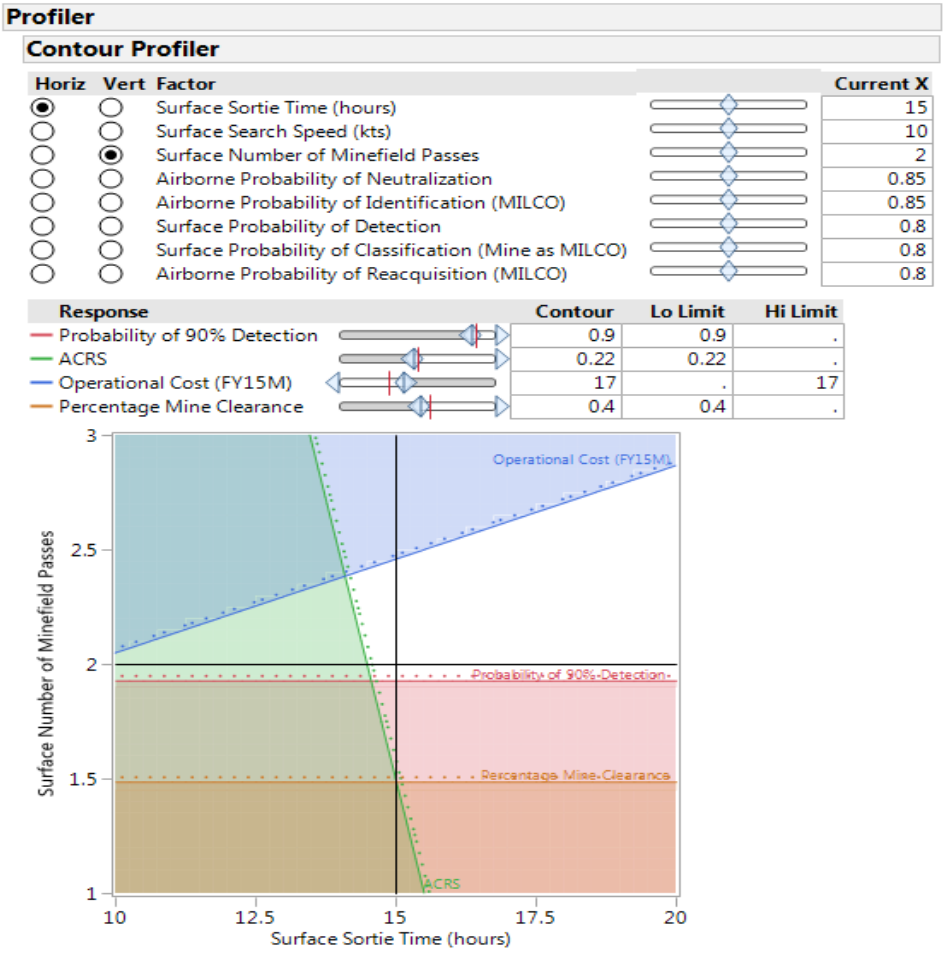
Figure 79 Operational Tradespace Visualization (View 1): LCS



The LCS configurations will not be discussed in the same level of detail because many of the conclusions are similar, but the same approach used to explore the MCM-1 configuration tradespaces can be used to examine the LCS configuration tradespaces. For example, the LCS system requires a Probability of Detection of at least approximately 0.78 to ensure that the threshold of 90% Probability of 90% Detection is met. Once again, a single minefield pass is infeasible for all values of Probability of Detection from an operational perspective and a third minefield pass is infeasible for all values of Probability of Detection from a cost perspective. An alternative projection shows the

tradespace between the Surface Sortie Time, on the x-axis, and the Number of Minefield Passes (Figure 80).

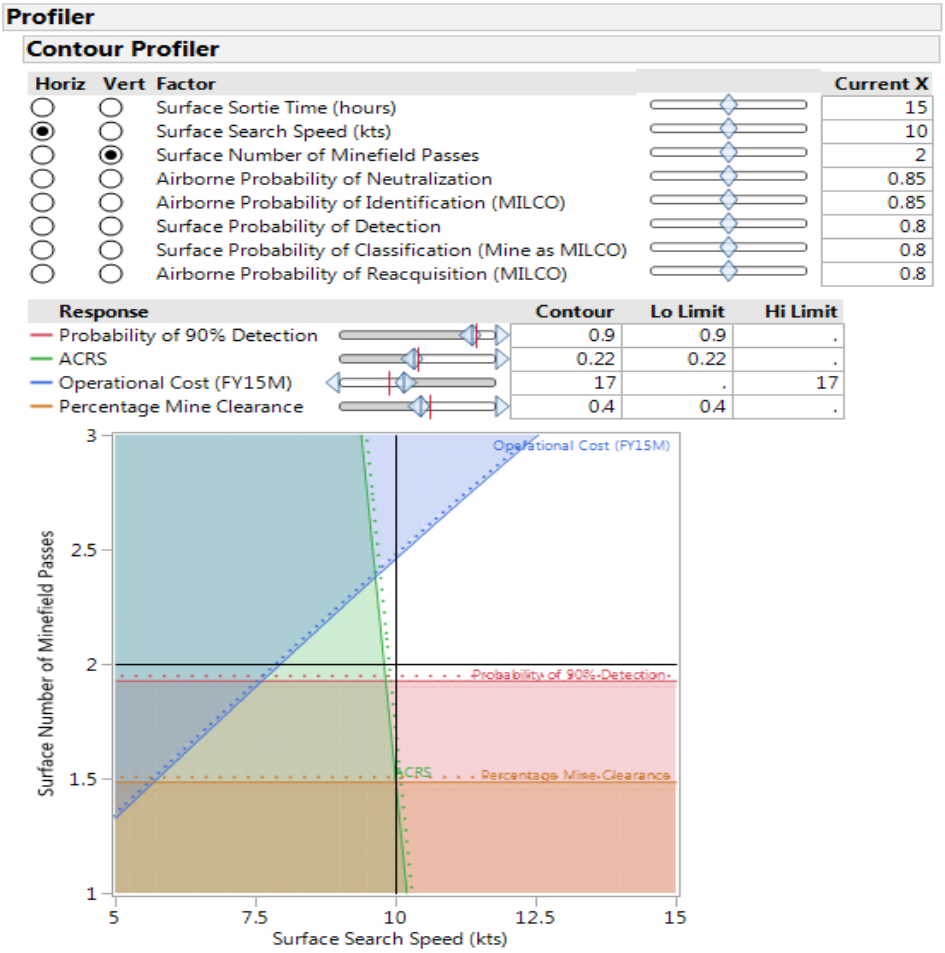
Figure 80 Operational Tradespace Visualization (View 2): LCS



As with the previous two-dimensional projection, the threshold for the Time to Achieve 90% Detection and for Percentage Mine Clearance eliminate all configurations where only a single minefield pass is conducted and the threshold for Operational Cost eliminates all configurations where three minefield passes are conducted. Considered with the ACRS threshold, two minefield passes are required and a Surface Sortie Time of approximately 15 hours is required. A similar situation is shown in Figure 81, which

presents a two-dimensional tradespace projection with the Number of Minefield Passes on the y-axis and the Surface Search Speed on the x-axis.

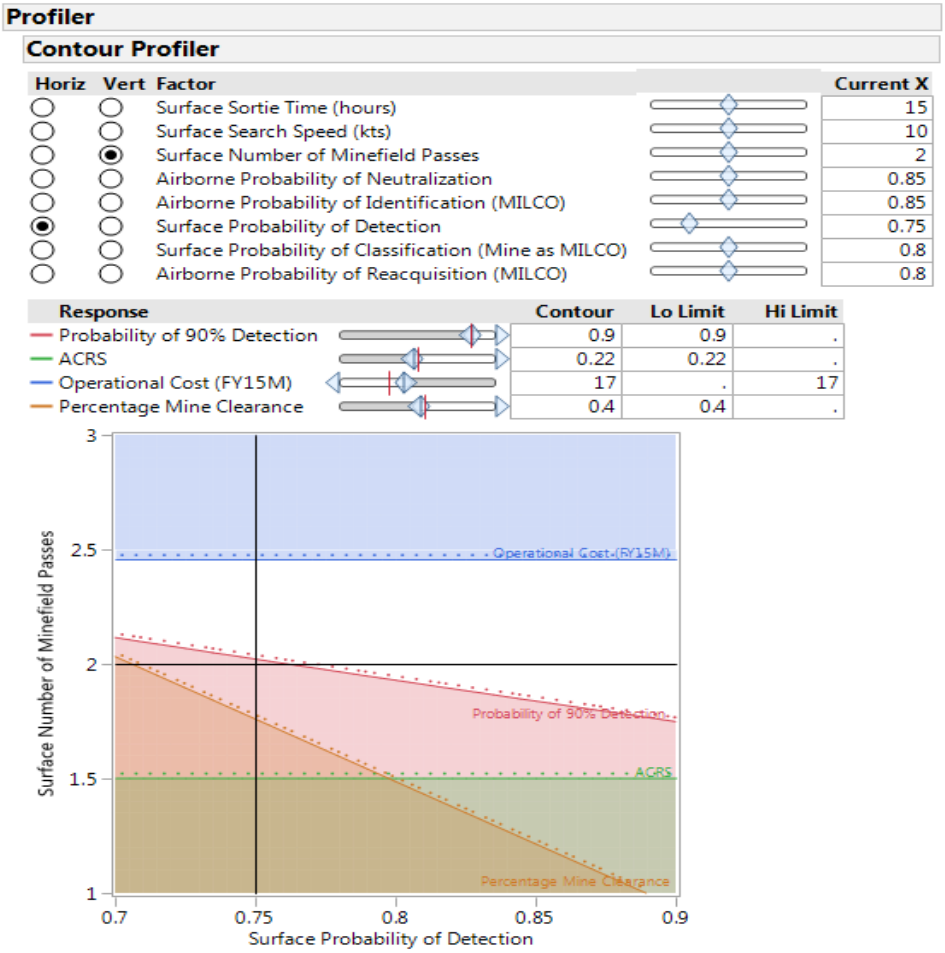
Figure 81 Operational Tradespace Visualization (View 3): LCS



Once again, the operational MOEs restrict all combinations with a single minefield pass. The Operational Cost threshold suggests that a third minefield pass is possible is the Surface Search Speed exceeds 12 knots. Once again, the information obtained from this visualization can be used to inform operational decisions if the Probability of Detection is restricted. Figure 82 presents a visualization of the two-dimensional projection with the Number of Minefield Passes shown on the x-axis and the

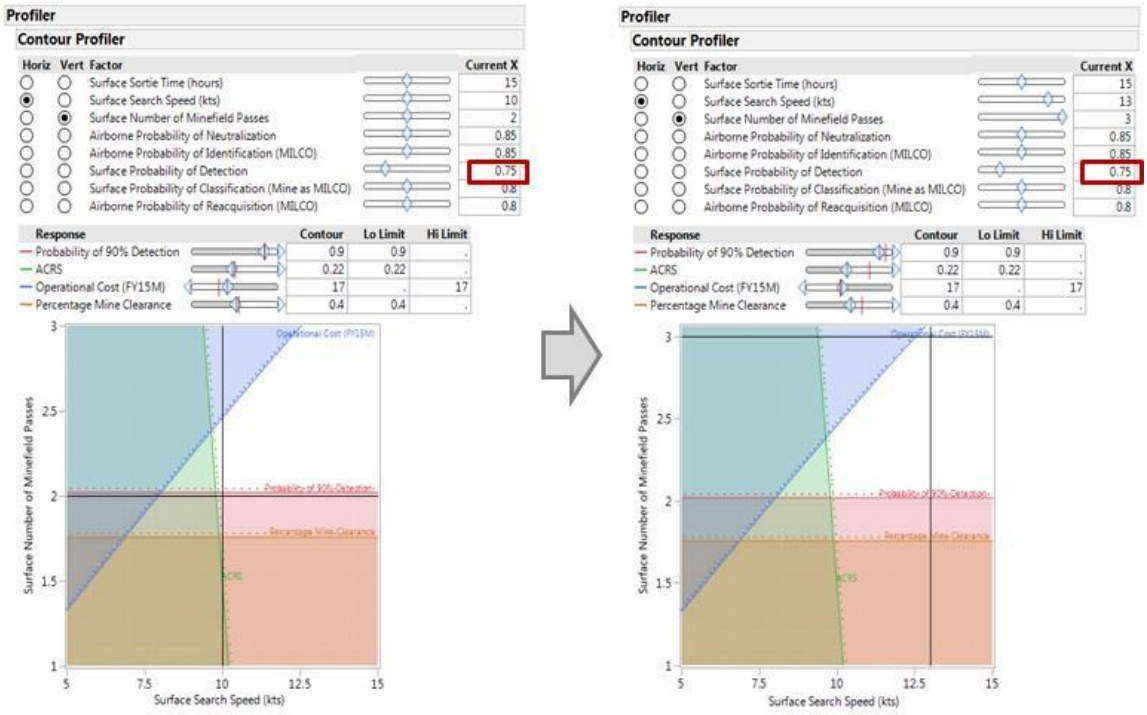
Probability of Detection shown on the x-axis with the Probability of Detection restricted to 0.75.

Figure 82 Operational Tradespace Visualization (View 4): LCS



Notice that the system configuration prescribed by the crosshair is identified as infeasible. If the Probability of Detection cannot exceed 0.75 it is necessary to conduct a third minefield pass to satisfy the Probability of 90% Detection threshold. Recalling the information presented in Figure 81 it may be possible to conduct a third minefield pass if the Surface Search Speed is increased. Figure 83 presents a visualization of the implementation of such a decision.

Figure 83 Operational Tradespace Visualization (View 5): LCS



Note that the system configuration specified on the left of Figure 83 is infeasible for the Probability of 90% Detection due to the restriction of the Probability of Detection to 0.75. Once again a potential solution is highlighted on the right of Figure 83. By increasing the Surface Search Speed from 10 knots to 13 knots, it is now possible to conduct a third minefield pass without violating the Operational Cost threshold. In turn this allows the system to satisfy the Probability of 90% Detection threshold even though the Probability of Detection has been restricted to 0.75.

The presentation of the examination of the operational and system tradespace for each of the MCM system configurations is intended to emphasize the importance of complete tradespace exploration through a dynamic tool. The MBSE MEASA relies on development of external models and simulations, based on system architecture products. Detailed analysis of those external models (in this case an operational simulation model) enables development of predictive surrogate models, which can subsequently be implemented in profilers to provide visualization of the system tradespace. Such an approach is intended to illuminate the full range of potential system design parameter

options, subject to operational MOE standards. The visualization is intended to aid decision making, rather than recommend a single system alternative. Recall that there are parallel efforts into efficient explorations of multi-attribute tradespaces and integration of this general approach with that work should be investigated. A more general integration of the MBSE MEASA advocated tradespace visualization with alternative response surface methods or subset selection procedures is also a worthy research direction.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

A. SUMMARY

This dissertation presents an MBSE MEASA that formalizes a comprehensive linkage between the system architecture domain and the system analysis domain. Due to the substantial expertise required to conduct research in each domain, recent developments focus largely within each of these domains, and there is insufficient emphasis on the link between descriptive architecture products, in particular SysML products, and external models and simulations. In particular, there is a need for a methodology that emphasizes that system architecture products should be the basis for not only physical system models, but also operational models and cost models. Further, those models must be capable of considering system design variables, operational variables, and environmental variables. The MBSE MEASA presents a revised approach for the utilization of SysML products to support external modeling and simulation efforts, groups those SysML products according to the traditionally conducted systems engineering processes, and demonstrates the utility of the new MBSE MEASA through a study of the MCM-1 Avenger and the LCS in an Active, Defensive MCM operation. The MBSE MEASA also provides a comprehensive demonstration of the methodology iteration, which provides a more explicit, dynamic iteration capability than possible using any alternative MBSE methodology.

Recall that this research is motivated by the need to produce more complete system requirements. This research presents a procedure that begins with an initial set of system requirements, translates those requirements into detailed SysML architecture products, uses those SysML architecture products as the foundation for assessment of system performance through designed experiments of external models and simulations, and uses the results of those assessments to visualize the impact that each system design parameter has on the operational performance of the system. This identifies feasible set of system design parameters that may be investigated in more detail, as well as identifies those system requirements that have little to no impact on system performance and therefore may not require additional analysis or definition. Most importantly, adherence

to this methodology provides traceability from an initial set of requirements to a set of architecture products and external models and simulations that can be used to assess those requirements and develop system design parameters. Finally, because the MBSE MEASA explicitly considers design, operational, and environmental variables, the MBSE MEASA uniquely describes how impactful variables in each of these domains (as well as potential interactions between those variables) can be integrated into future iterations of system architecture products.

The dissertation expands the scope of the existing MBSE methodologies developed by IBM, NASA, INCOSE, Vitech, etc. The dissertation extends the current focus of MBSE by expanding the focus from descriptive architectural based frameworks to a more comprehensive framework that links formally defined architecture products to detailed external models and simulations. While the MBSE MEASA has a broader applicability than any existing MBSE methodology, it is also vitally important to position the MBSE MEASA in relation to recent academic and professional research, particularly in two areas: model-based systems engineering and simulation analysis.

This research is broader in scope and applicability than existing model-based systems engineering focused research. Recent work, such as Wang and Dagli (2008), Ge, Hipel, Yang, and Chen (2013), and Kim, Fried, Menegay, Soremekun, and Oster (2013) present approaches for the automated execution of system analysis, through colored petri nets and discrete event models. However, these approaches assume that the system architecture is completely defined and static, and are currently restricted to systems whose operational procedure will not be altered. The MBSE MEASA makes no such assumptions, and is therefore applicable to a wider range of potential systems. Note that while this extension by the MBSE MEASA is currently relevant, it is not a general criticism of the idea of executable architectures and future coordination between the efforts may be possible. Limitations regarding computing power necessarily limit current implementations of executable architectures. However, in the future, some of the fundamental concepts developed in the domain of executable architectures may be integrated with the emphasis on designed experiments presented in the MBSE MEASA

as a means to automate the translation of SysML diagrams to more detailed external models.

While MBSE is a relatively new field, modeling and simulation, in particular large scale modeling and simulation, has existed for many years and has successfully supported system development. Traditional approaches to model development (such as the assumptions document, presented in Law (2014) or conceptual models, presented in Sargent (2013)) share many goals and characteristics with descriptive architectural based development. While there are similar concepts, the goal of the MBSE MEASA is to provide a more powerful framework to link descriptive and analysis focused models. The MBSE MEASA establishes a framework that ensures traceability and consistency between multiple models in an easier-to-manage environment than previously possible. The MBSE MEASA defines an approach that mandates and enforces this consistency and facilitates identification and resolution of conflicts for system requirements, functions, and physical elements.

Finally, the MBSE MEASA utilizes SysML products as a basis for system architecture development to ensure compatibility with the widest range of MBSE approaches. While SysML is a relatively new approach for system description and development, several recent research efforts demonstrate the potential utility of SysML focused development. Of particular note, Johnson (2008), Cao, Liu, and Paredis (2011), Qamar, During, and Wikander (2009), Palachi, Cohen, and Takahaski (2013) and Spangelo, et al. (2013) all demonstrate the potential for automated generation of physical models based on SysML products. Huang, Ramamurthy, and McGinnis (2007), Huang (2011), and Bataresh and McGinnis (2012) also demonstrate a similar approach for the generation of manufacturing models based on SysML products. That research is a tremendously powerful generation of the potential utility of SysML. The MBSE MEASA presents a more general framework for the overall utility of SysML products that integrates those approaches, at the same time that it emphasizes the need to consider large number of system parameters, system component interactions, and system operational and environmental interactions, considerations which are limitations of existing works into the generation of external models using SysML products. Once again, this is not a

universal criticism of existing work, the assumptions and limitations associated with the automated SysML-focused research and executable architecture research exist because it is necessary to demonstrate simple use cases before more complicated cases.

The MBSE MEASA establishes a framework usable as the basis for future developments in MBSE focused research. Ideally, the lessons learned and computational advances made possible by existing work in executable architecture research and automated SysML development research will integrate within the framework of the MBSE MEASA to support system development in a rapid fashion, ensuring a more holistic approach to system development.

The MBSE MEASA uses SysML products as a basis to ensure the usability of the methodology. SysML, the current focus of a large portion of current MBSE research, facilitates implementation in conjunction within any existing MBSE methodology. This research focuses on the analysis portion of MBSE and presents a methodology that provides a roadmap for any user to leverage a set of SysML architecture products, which are defined in this research and prescribed by almost every existing MBSE methodology, to conduct detailed analysis of system performance and behavior through external models and simulations. This improves the efficiency and effectiveness of the engineering process by linking detailed system architecture products to detailed external models and simulations. This improves traceability, later iterations of those architecture products, and facilitates assessment of the quality of previously defined system requirements.

This research demonstrates the potential utility of the MBSE MEASA through analysis of an Active, Defensive MCM operation. This demonstration establishes a roadmap to implementation of the methodology for any future user. However, it is also useful to refer to the previously presented intended characteristics of a systems engineering process to ensure that the MBSE MEASA supports each of those characteristics.

1. The process must be comprehensive. It must not focus on individual aspects of the system and instead should consider the system as an integrated whole.

By leveraging SysML products, which capture not only system structures and functions, but also the relationships between system elements, potential constraints on each system element, and the allocations of system components to system functions, the MBSE MEASA ensures that architectural representations of the system consider the system as an integrated whole. Furthermore, the MBSE MEASA's use of those products to define the behaviors and physical entities that must be represented in any external model or simulation ensures that all aspects of the system are included and assessed to determine their impact on system performance, behavior, structure, and cost.

2. The process must be iterative. It must be capable of considering an initially stated operational need and evaluating system configurations against that need, while simultaneously scoping the operational capabilities of the system such that the process can be repeated for a more focused operational need.

The MBSE MEASA supports analysis within some implementation of the systems engineering methodology or within some implementation of an MBSE methodology. It defines a path for translating a set of system requirements into system architecture products, which facilitate development of external models and simulations. Modeling and simulation analysis results are the basis for the development of a tradespace visualization tool. Properly visualizing the system tradespace allows for definition of a feasible set of system configurations. The MBSE MEASA explicitly defines an iteration procedure based on that tradespace analysis to update system architecture products for impactful design, operational, or environmental variables (as well as any potential interactions between those variables).

3. The process must be defined by a logical sequence of activities and decisions. As noted, the process must be iterative, but there is necessarily an element of sequence. The process must explicitly define the ordering and characteristics of each event in the process. Ambiguity must be kept to a minimum in order to clearly delineate each event and clearly define the achievements that trigger the transition between events.

The MBSE MEASA defines the ordering and application of SysML based architecture products. This aligns the methodology with the current direction of MBSE research, which advocates SysML for its clear system architecture representation

capability. Furthermore, the MBSE MEASA prescribes a defined set of steps that facilitate the use of SysML products as a basis for external models and simulations.

4. The process must transform an operational need into a description of system performance parameters and preferred system configurations. This is perhaps the most important characteristic of a quality systems engineering process. In short, the objective of any systems engineering process is to ensure that the decisions that lead to recommendation of a system configuration can be directly linked to a clearly defined operational need.

As mentioned, the intended output of the MBSE MEASA is a definition of a feasible set of system configurations. Adherence to the MBSE MEASA ensures that the feasible set of system configurations is traceable to a set of system functions and requirements through SysML architecture products. That traceability ensures that there is no disconnect between the originally identified stakeholder need and the final set of feasible system configurations identified by the MBSE MEASA.

Finally, it is useful to recall the intended benefits of MSBE developed by Friedenthal, Griego, and Sampson (2007) and show that the MBSE MEASA aids realization of those benefits.

1. Improved communications among the development stakeholders (Friedenthal, Griego, and Sampson (2007, 7).

    a. The MBSE MEASA uses stakeholder input to develop a SysML Requirement Diagram, which is the simplest way to capture stakeholder needs in a defined, concise format. Because this Requirement Diagram is used as the basis for architecture construction (and therefore model building) it can easily be updated based on the results of the MBSE MEASA. For example, the tradespace examination in Chapter IV recommended a Probability of Detection for the LCS MCM system of at least 0.80 (recall that this exploration and the associated recommendations apply exclusively to the LCS MCM system defined by constant values for each of the other factors shown in Figure 83). Figure 34 presented a Requirement Diagram for Minehunting Operation, which presents the

Probability of Detection as a system requirement. This requirement can be updated based on the results of the MBSE MEASA and any stakeholder discussions can revolve around model results that are traceable back to the original Requirement Diagram.

2. Increased ability to manage system complexity by enabling a system model to be viewed from multiple perspectives, and to analyze the impact of changes (Friedenthal, Griego, and Sampson (2007, 7).

    a. One of the central goals of this research is to develop an analysis methodology that supports the development of architecture models and external operational simulation models for large scale, complex systems. Accordingly, SysML products, which are currently the most popular tool for development of system architecture products, are used to ensure that a comprehensive system model is developed that allows the system to be viewed from multiple perspectives. This facilitates development of external simulation models that are traceable and establishes a linkage between any proposed system design changes to originally established system requirements (and therefore to an original set of stakeholder needs).

3. Improved product quality by providing an unambiguous and precise model of the system that can be evaluated for consistency, correctness, and completeness (Friedenthal, Griego, and Sampson (2007, 7).

    a. As mentioned in the discussion of the utilization of the MBSE MEASA to facilitate multi-perspective system views, the utilization of SysML products as a baseline for system architecture and system analysis ensures that the full set of system architecture products can be evaluated for completeness and consistency. If some expected system functionality is not present in an external operational simulation, the accuracy and completeness of the SysML Activity Diagram and the SysML Sequence Diagram can be evaluated and updated to properly define the sequencing

of the expected functionality within the simulation model and to describe the expected system components that are required to conduct the activities. If some expected system component is not included in a cost or physical model, the SysML Block Definition Diagram can be examined to determine whether or not the component is currently considered a part of the system physical hierarchy and, if it is not, the SysML Internal Block Diagram can be examined to determine what system components are performing the activities expected to be performed by the missing component. The comprehensive, unambiguous nature of these architecture models ensures that any external models built to support system analysis can be evaluated and revised to ensure that they provide a complete, correct, consistent representation of each system component and each system behavior.

4. Enhanced knowledge capture and reuse of information by capturing information in more standardized ways and leveraging built in abstraction mechanisms inherent in model driven approaches. This in turn can result in reduced cycle time and lower maintenance costs to modify the design (Friedenthal, Griego, and Sampson (2007, 7).

    a. The MBSE MEASA provides a formal definition of the use of SysML products to establish a linkage between the system architecture and system analysis domain. This definition establishes a bridge between the domains where standardized information can be shared to remove any potential conflicts between architecture models and analysis models (either operational models, physical models, or cost models). As with any proposed method of operation the utility of the MBSE MEASA will certainly be a function of proper implementation, however creation of a standard set of products that facilitate communication between multiple domains should reduce the potential for conflict and therefore reduce the time needed to rework system architecture models to reflect external models and to reduce the time needed to revise system operational,

physical, and cost models to reflect changes to system functions and components that results from alterations to system architecture models.

B. CONCLUSIONS

The MBSE MEASA developed in this research strengthens the linkage between descriptive system architecture products and system analysis products. The current direction of MBSE research suggests that SysML products will be the standard for system architecture development for the foreseeable future. Accordingly, this research defined an analysis methodology that leverages SysML products but expands their utility by defining a comprehensive framework for their application to external models. Definition of a procedure for using those SysML products to support the development and structured exploration of external models and simulations is a valuable approach within the system architecture domain and system analysis domain. This research maximizes the utility of descriptive system architecture products by defining a method for utilizing those products to evaluate the operational effectiveness, structure, and cost of potential system configurations. This research emphasizes that the results of those external models and simulations must assess any previously established system requirements. This assessment ensures that the set of system requirements completely describe a system that is feasible and effective in terms of operation, structure, and cost. Because this comprehensive framework links descriptive system architecture products to detailed system analysis products, this research is able to develop a unique iteration procedure that demonstrates proper integration of analysis results into future versions of descriptive architecture products.

C. AREAS TO CONDUCT FUTURE RESEARCH

This research presented a defined methodology for linking the system architecture and system analysis domains in the context of model-based systems engineering. There exist numerous potential related research areas that would further extend the systems engineering body of knowledge. The most direct contribution consists of applications of the MSBE MEASA to non-traditional systems (systems with limited control over design as well as systems that exhibit emergent behavior are potential examples, although others may exist).

Another logical (and potentially related) application of the MSBE MEASA is to development of systems of systems. While this methodology was demonstrated using an integrated set of systems in the MCM simulation, this only establishes utility for a "directed" system of systems, or one where the command and control of the set of systems can be attributed to a single user (or set of users) and each of the systems is designed and operated to satisfy a predetermined set of functions. Other systems of systems (acknowledged, collaborative, and virtual systems of systems) are often distributed, independently managed and operated, and may not operate in support of the same defined set of functions. This certainly introduces new challenges due to the potential for emergent behaviors, the lack of central ownership and management, the potential for potentially conflicting objectives, and the inability to define a unifying set of standards and goals. In particular, research and applications in this area may benefit from further investigation of heuristics or modeling techniques that allow for mapping of operational simulation inputs to system synthesis inputs.

APPENDIX A. MBSE MEASA COMPARISON TABLE

This appendix presents a detailed comparison table that positions the MBSE MEASA in terms of recent work. The table presents general criteria in six areas: Architectural Approach, External Modeling Approach, External Model Components, Analysis Approach, Application & Demonstration, and Iteration. It summarizes the contributions that each of the leading MBSE Methodologies, recent work in MBSE Development, and relevant work in Simulation & Analysis made to each of those areas. The table is presented in three parts to facilitate readability.

Table 10 MBSE MEASA Comparison Table (Part 1: MBSE MEASA and MBSE Methodologies)

		MBSE MEASA	IBM Harmony for Systems Engineering	INCOSE Object Oriented Systems Engineering Method	Vitech Model Based Systems Engineering Method	NASA Jet Propulsion Lab State Analysis	Dori Object Process Methodology	Weilkiens Systems Modeling Process
		MBSE Methodologies						
Architectural Approach	Use of SysML	X	X	X	X			X
	Requirements Diagram	X	X	X	X			X
	Functional Architecture	X	X	X	X			X
	Physical Architecture	X	X	X	X			X
	Parametric Diagram		X	X	X			X
	Alternative Architecture Approach				X	X	X	
External Modeling Approach	Modeling Restricted to Architecture			X				X
	Operational Models	X	X		X			
	System Models	X	X			X	X	
	External Operational Simulation Model	X			X			
	Parametric/Spreadsheet Models	X	X		X	X	X	
External Model Components	Design Variables	X	X		X	X	X	
	Operational Variables	X	X				X	
	Environmental Variables	X	X					
Analysis Approach	Analysis of Alternatives	X	X	X	X	X	X	X
	Analysis Through Value Functions		X	X	X	X	X	X
	Large Number of System Components	X						
	Design Interactions	X						
	Operational Interactions	X						
	Environmental Interactions	X						
	Defined Experimental Design	X						
	Development of Metamodels	X						
	Metamodels for Multiple MOEs	X						
	Single Design Recommendation		X	X	X	X	X	X
	Robust Design Methods							
	Tradespace Exploration	X						
Application & Demonstration	Deterministic Models	X			X			
	Stochastic Models	X						
	Defense Systems	X						
Iteration	Discussed Iteration	X	X	X	X	X	X	X
	Demonstrated Iteration	X					X	
	Applied to SysML	X						

Table 11 MBSE MEASA Comparison Table (Part 2: Recent MBSE Development)

		Acheson, Dagli, Kilicay-Ergin (2013)	Balestrini-Robinson, Freeman, Browne (2015)	Batarseh, McGinnis (2012)	Blackmann, Sarkani, Mazzuchi (2013)	Cao, Liu, Fan, Fan (2015)	Carson, Sheehy (2011)	Fisher (2013)	Giammarco, Auguston (2013)	Hanemann, Rommma (2015)	Huang (2011)	Huang, Ramanathan, McGinnis (2007)	Johnson (2008)	Kenley, Dannenhoffer, Wood, DeLaurentis (2014)	Kim, Fried, Menassey, Soremekun, Oster (2013)	Liston, Kobak, Dungan, Byrne, Young, Heavey (2011)	Neches, Madni (2013)	Palachi, Cohen, Takashi (2013)	Qamar, Dunlap, Wikander (2008)	Ross (2003)	Ross, Stein, Hastings (2014)	Russell (2012)	Ryan, Sarkani, Mazzuchi (2013)	Sitterle, Freeman, Goerger, Ender (2015)	Spangelo, Cutler, Anderson (2013)	Summers, Eckert, Goel (2013)	Tolk, Hughes (2014)	Wang, Dagli (2011)	Wu, Ciarallo, Geelhenson (2013)
Architectural Approach	Use of SysML	X	X	X	X	X	X	X	X		X	X	X	X	X	X		X	X			X			X		X	X	X
	Requirements Diagram	X	X	X	X	X	X	X	X		X	X	X	X	X	X		X	X			X			X		X	X	X
	Functional Architecture	X	X	X	X	X	X	X	X		X	X	X	X	X	X		X	X			X			X		X	X	X
	Physical Architecture	X	X	X	X	X	X	X	X		X	X	X	X	X	X		X	X			X			X		X	X	X
	Parametric Diagram	X	X	X	X	X	X	X	X		X	X	X	X	X	X		X	X			X			X		X	X	X
	Alternative Architecture Approach							X	X	X	X						X			X	X		X	X		X			
External Modeling Approach	Modeling Restricted to Architecture				X				X					X		X						X							X
	Operational Models			X				X		X	X		X				X			X	X		X				X	X	
	System Models					X	X		X	X		X	X		X		X	X	X						X		X		
	External Operational Simulation Model	X		X			X			X	X						X										X	X	
	Parametric/Spreadsheet Models				X	X	X	X	X	X			X			X				X	X		X	X			X		
External Model Components	Design Variables	X		X	X	X	X	X	X	X	X	X	X	X		X			X	X	X	X		X		X		X	X	
	Operational Variables	X					X	X		X		X	X						X	X		X		X		X		X	X	
	Environmental Variables				X			X		X			X									X							X	X	
Analysis Approach	Analysis of Alternatives				X	X	X	X	X	X	X	X	X	X		X					X	X		X	X	X		X	X	
	Analysis Through Value Functions				X	X		X	X			X										X		X	X					
	Large Number of System Components																													
	Design Interactions					X	X	X	X	X		X	X	X		X			X	X		X		X				X		
	Operational Interactions						X	X		X			X			X			X	X		X						X		
	Environmental Interactions				X			X		X			X									X						X		
	Defined Experimental Design																													
	Development of Metamodels																								X			X		
	Metamodels for Multiple MOEs																								X			X		
	Single Design Recommendation				X	X	X				X			X					X				X		X		X	X	
	Robust Design Methods																													
	Tradespace Exploration																X			X	X				X			X		
Application & Demonstration	Deterministic Models				X	X		X			X	X		X				X		X	X		X					X	X	
	Stochastic Models	X					X			X	X				X			X			X							X	X	
	Defense Systems				X					X															X					
Iteration	Discussed Iteration				X							X			X				X											
	Demonstrated Iteration																													
	Applied to SysML																													

Table 12 MBSE MEASA Comparison Table (Part 3: Relevant Simulation & Analysis Development)

		Ashpari (2012)	Becker, Byram, Frank, Hatten, Kim, Miller, Myers, Schonhoff, Whitehouse (2014)	Ge, Hipel, Yang, Chen (2013)	Hunman, Madni (2014)	Kaymal (2013)	Kleijnen, Sanchez, Lucas, Cioppa (2005)	Law (2009)	Lucas, Ketcha, Sanchez, Sanchez, Anderson (2015)	MacCalman, Kwak, McDonald, Upton, Snider, Hill, Wood, Eversteijn (2015)	MacCalman, Kwak, McDonald, Upton (2015)	MacCalman, Reevy, Paulo (2016)	McKeown (2013)	Parker (2015)	Sanchez, Wan (2012)	Sanchez (2000)	Sargent (2013)	Shi, Le, Pinchal (2011)	Siebers, Macal, Garnett, Buxton, Pidd (2010)	Spero, Avera, Valdez, Goerger (2014)	Tremi (2013)	Wakeman (2012)	Yassin (2012)
Architectural Approach	Use of SysML									X	X							X					
	Requirements Diagram									X	X							X					
	Functional Architecture									X	X							X					
	Physical Architecture									X	X							X					
	Parametric Diagram									X	X							X					
	Alternative Architecture Approach		X	X								X								X			
External Modeling Approach	Modeling Restricted to Architecture																X						
	Operational Models	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	System Models											X					X			X			
	External Operational Simulation Model		X		X	X	X	X		X	X	X	X	X	X	X		X	X		X	X	X
	Parametric/Spreadsheet Models	X		X				X	X			X			X								
External Model Components	Design Variables	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X		X	X	X	X
	Operational Variables	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X		X	X	X	X
	Environmental Variables	X	X			X	X	X	X	X	X	X	X	X	X	X		X		X	X	X	X
Analysis Approach	Analysis of Alternatives	X	X	X	X	X	X		X	X	X	X	X	X	X	X					X	X	X
	Analysis Through Value Functions			X											X	X							
	Large Number of System Components	X	X			X			X	X			X	X	X	X					X	X	X
	Design Interactions	X	X		X	X	X		X	X	X	X	X	X	X	X					X	X	X
	Operational Interactions	X	X		X	X	X		X	X	X	X	X	X	X	X					X	X	X
	Environmental Interactions	X	X			X	X		X	X	X	X	X	X	X	X					X	X	X
	Defined Experimental Design	X	X			X	X		X	X	X	X	X	X	X	X					X	X	X
	Development of Metamodels	X	X			X	X		X	X	X	X	X	X	X	X					X	X	X
	Metamodels for Multiple MOEs		X				X		X	X	X	X	X	X	X	X					X	X	X
	Single Design Recommendation		X	X																			
	Robust Design Methods						X	X	X							X							
	Tradespace Exploration	X	X			X	X		X	X	X	X	X	X						X	X	X	X
Application & Demonstration	Deterministic Models	X			X									X	X		X						
	Stochastic Models		X		X	X	X		X	X	X	X	X	X	X	X		X			X	X	X
	Defense Systems	X	X			X	X			X	X	X	X	X						X	X		X
Iteration	Discussed Iteration						X								X								
	Demonstrated Iteration						X																
	Applied to SysML																						

APPENDIX B. EXPERIMENTAL DESIGN VERSUS BASELINE FOLLOWED BY EXCURSIONS

This appendix provides guidance in two areas. First, it demonstrates the risks associated with testing systems by establishing a baseline and conducting individual excursions, and shows that proper experimental design utilization prevents mistakes in test configuration specification that may result from testing by "baseline followed by excursions." Second, it demonstrates that the types of experimental designs that may be familiar to systems engineers from experience with physical system testing may experience limitations when used for simulation models and presents guidance regarding the selection of efficient experimental designs.

As noted previously, Friedenthal, Griego, and Sampson (2007) state that MBSE provides five major benefits, summarized as: improved communications, increased ability to manage system complexity, improved product quality, enhanced knowledge capture and reuse of information, and improved ability to teach and learn systems engineering fundamentals. Experimental design, particularly in the context of simulation experiments, is vitally important to realizing several of those benefits. Increased ability to manage system complexity cannot be achieved without development of a system architecture model that can be viewed from many perspectives to examine the impact of those potential changes. Experimental design specifies the system configurations that should be modeled in order to properly analyze the impact of changes in system configurations on system performance. Improved product quality cannot be achieved without capturing information in standardized ways. Again, experimental design provides the standards for simulation model construction that ensures that all product decisions are made in support of the end goal of increased system performance. As mentioned in Chapter III, several excellent references, in particular Montgomery (2012) provide a comprehensive overview of experimental design. Sanchez et al. (2012) present a more specific overview of experimental design for simulation experiments. Any user of the MBSE MEASA would be well served to review that work; however, a brief discussion of experimental design clarifies the benefits that experimental design may have in the

context of MBSE. This provides a basic introduction and to discourages users from implementing a "baseline followed by excursions" approach to system testing.

A. PRINCIPLES OF EXPERIMENTAL DESIGN

Before examining specific experimental designs techniques in detail, it is useful to consider the general purpose of DOE. Experimental designs are the basis for conducting tests and experiments. In the context of systems engineering, the purpose of conducting tests or experiments is to understand the drivers of a system's performance. Kleijnen et al. (2005) present three basic goals of simulation analysis (these goals also apply to systems engineering tests and experiments): "developing a basic understanding of a particular simulation model or system, finding robust decisions or policies, and comparing the merits of various decisions or policies." The first goal is most applicable to tests and experiments for the types of large scale, complex systems being studied by this research. Kleijnen at al. (2005) further specify that these tests or experiments may be conducted "to gain insight into situations where the underlying mechanisms are not well understood, and where real-world data are limited or even nonexistent." This serves as a useful definition of the purpose of the tests and experiments relevant to this research. This establishes that, in general, a test or experiment is used to establish a relationship between input variables, which characterize system capabilities or configurations, and output variables, which characterize system performance.

Experimental designs add rigor to the process of experimentation by planning the experiment and defining the nature of the data to be collected. This allows experimenters to effectively conduct tests and experiments that uncover insights regarding system performance. As mentioned previously, the often used "baseline followed by excursions" approach may lead to inappropriate conclusions or an incomplete understanding of the true drivers of system performance. Use of a good experimental design ensures that the assumptions behind any statistical tests conducted on the experimental results are not violated.

At the simplest level, a conceptual model of a stochastic relationship between factors (inputs) and responses (outputs) is:

$$Y = f(x) + \varepsilon \tag{1}$$

A simple functional model that is often assumed, in practice, to represent this function in the basis linear model:

$$y_i = \beta_0 + \beta_1 x_1 + \varepsilon \tag{2}$$

where y_i is the value of some output variable, β_0 is a constant intercept value, β_1 is a multiplying coefficient for x_1, x_1 is the value of some input, and ε is an error term. This is especially useful when attempting to predict system performance. By formulating the relationship between inputs and output in this fashion, it is possible to predict the change in system performance (y_i) associated with a change in some system characteristic (x_1). An instructive example is an equation that quantifies how an increase in the amount of fertilizer used in farming (x_i) impacts the total crop yield (y_i). Data collected either from a designed experiment or from observation can be used to estimate the coefficients of the model.

Unfortunately, most processes are not as simple as in the example above. Typically, more than one system characteristic (input) will impact system performance (output). Accordingly, the type of relationship described above becomes more complex. Examples include:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon \tag{3}$$

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \varepsilon \tag{4}$$

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \varepsilon \tag{5}$$

Equation (2) previously presented system performance in terms of a single input variable, Equations 3–5 present system performance as a function of: (3) two input variables; (4) the linear and quadratic effect of one input variable; (5) two input variables as well as the interaction between those variables. Examples of these equations are fairly intuitive. A system with behavior specified by Equation (5) provides an excellent example of the value of experimental design.

Equation (5) describes a situation where the behavior of the system cannot be understood through an isolated study of the system components. As an example, recall that two of the variables with the largest impact on LCS MCM performance were the Probability of Mine Classification and the Probability of Mine Neutralization. Accordingly, an engineer of LCS MCM subsystems (who suspected that these variables would have a significant impact on performance but did not have access to the detailed analysis presented earlier in this research) may be interested in examining the impact that the probability of classification and neutralization have on system performance. If the engineer were interested in utilizing a model (such as the one presented earlier in this research) to describe the impact of these variables very generally, the system configurations to be tested must be defined before testing begins. There are an infinite number of system configurations (in terms of probability of classification and probability of neutralization) that may be tested. For the purposes of this example, the probabilities of classification and neutralization are both restricted to the range [0.70, 0.90]. If the engineer were to proceed with testing through the "baseline followed by excursions" approach a "baseline" system configuration could be established with the probability of classification and probability of neutralization both set to a minimum value, in this case 0.70. The engineer could conduct a test at this baseline configuration and subsequently conduct follow on tests where first the probability of classification is maximized and second the probability of neutralization is maximized. Table 13 presents a definition of what these tests (which define system configurations) would look like in terms of probability of classification and probability of neutralization.

Table 13 Example Test Configurations: Baseline Followed by Excursions

Test #	Probability of Classification	Probability of Neutralization
1	0.7	0.7
2	0.7	0.9
3	0.9	0.7

The engineer could subsequently proceed to collect performance data for each of these test configurations. In terms of the LCS MCM model, the performance data of interest may be the percentage of mines successfully neutralized. Note that proper testing procedures dictate that each test configuration be replicated (tested multiple times) to enable examination of the variability associated with each test configuration, in this example 30 replications of each test configuration is presented. Equation (6) presents an example equation that describes the true system performance (where y_i represents the percent clearance, x_1 represents the probability of classification, and x_2 represents the probability of neutralization) and Table 14 presents an example of what the data collection looks like for a situation where the engineer conducts the three tests prescribed by the "baseline followed by excursions" approach (note that some variability was introduced to the model to emphasize the importance of replication).

$$y_i = 0.35x_1 + 0.35x_2 + 0.1x_1x_2 \tag{6}$$

Table 14 Example Test Data: Baseline Followed by Excursions

Test #	Probability of Classification	Probability of Neutralization	Percent Clearance (Test 1)	Percent Clearance (Test 2)	Percent Clearance (Test 3)		Percent Clearance (Test 28)	Percent Clearance (Test 29)	Percent Clearance (Test 30)
1	0.7	0.7	0.573	0.569	0.563		0.565	0.569	0.571
2	0.7	0.9	0.663	0.647	0.653		0.643	0.659	0.659
3	0.9	0.7	0.645	0.653	0.649		0.661	0.657	0.643

Subsequent to this data collection, the engineer may conduct regression analysis, which can be used to describe the performance of the system (the percent clearance) in terms of the input variables (in this case, the probability of classification and neutralization). As noted in Montgomery (2012), the use of regression models to present the results of an experiment or model is intuitive and, in this specific example, demonstrates that errors in the characterization of system behavior can result from a flawed approach to the specification of test configurations. The results of least squares regression based on the data presented in Table 14 are shown in Figure 84.

Figure 84 Example Regression Analysis: Testing With Baseline Followed by Excursions

Response Percent Clearance

Whole Model

Summary of Fit

RSquare	0.979628
RSquare Adj	0.97916
Root Mean Square Error	0.006111
Mean of Response	0.625244
Observations (or Sum Wgts)	90

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-0.052233	0.010496	-4.98	<.0001*
Probability of Classification	0.4463333	0.007889	56.58	<.0001*
Probability of Neutralization	0.4373333	0.007889	55.44	<.0001*

Sorted Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Probability of Classification	0.4463333	0.007889	56.58	<.0001*
Probability of Neutralization	0.4373333	0.007889	55.44	<.0001*

The analysis output of interest is highlighted in red in Figure 84. The coefficient associated with the probability of classification is estimated as 0.44 coefficient associated with the probability of neutralization is estimated as 0.43 (as well as an intercept value of -0.052) in the regression model. Recall that Equation (6) presented the true system performance (which regression analysis is attempting to estimate) and the coefficients associated with the probability of classification and neutralization were both 0.35. The regression analysis summarized in Figure 84 incorrectly estimated the relationship between the probabilities of classification and neutralization and the percent clearance. This incorrect estimation is not a result of incorrect regression analysis; rather it is a result of an incorrect definition of test configurations. By defining test configurations haphazardly, the engineer made it impossible to correctly describe the relationship between input variables and output variables.

A simple experimental design can be used to better define the test configurations that should be examined in this example. Recall that the engineer previously restricted the range of both the probability of classification and the probability of neutralization to

[0.70, 0.90]. To use experimental design generating software (or to generate a good experimental design by hand), the engineer must also specify the number of levels at which each input variable will be tested. For this example, assume that the engineer decides to test each variable at two levels (testing at the minimum and the maximum). The resulting test configurations are presented in Table 15.

Table 15 Example Test Configurations: 2 Variable, 2 Level Factorial Design

Test #	Probability of Classification	Probability of Neutralization
1	0.7	0.7
2	0.7	0.9
3	0.9	0.7
4	0.9	0.9

Table 15 presents an example of a two variable, two level factorial design (two levels indicating that each input can take only two values, in this case the minimum and maximum probabilities of classification and neutralization). The design provides the engineer with a list of test configurations that should be run and defines the value of each input variable for each of those tests. The engineer can subsequently proceed to collect output data (in this case percent clearance data) for each test configuration, the results are shown in Table 16. Note that once again 30 tests are conducted for each test configuration.

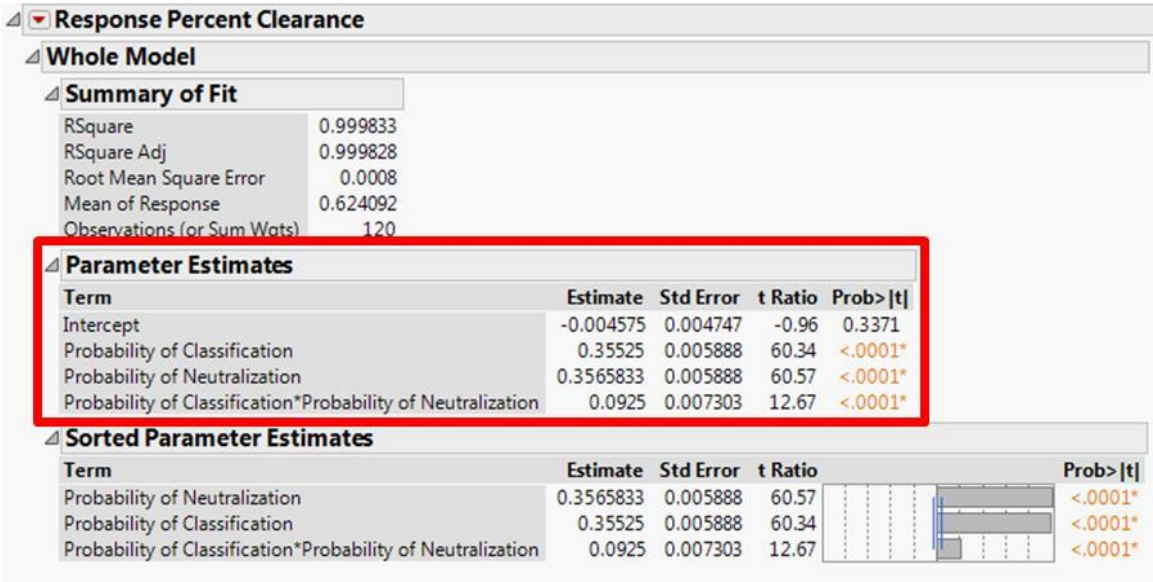
Table 16 Example Test Data: 2 Variable, 2 Level Factorial Design

Test #	Probability of Classification	Probability of Neutralization	Percent Clearance (Test 1)	Percent Clearance (Test 2)	Percent Clearance (Test 3)		Percent Clearance (Test 28)	Percent Clearance (Test 29)	Percent Clearance (Test 30)
1	0.7	0.7	0.557	0.519	0.543		0.525	0.537	0.529
2	0.7	0.9	0.629	0.617	0.643		0.635	0.629	0.623
3	0.9	0.7	0.623	0.631	0.641		0.629	0.633	0.607
4	0.9	0.9	0.699	0.721	0.701		0.713	0.705	0.707

This data can be analyzed using regression analysis to estimate the relationship between the input variables (the probabilities of classification and neutralization) and the

output variable (the percent clearance). The results of the regression analysis are shown in Figure 85.

Figure 85 Example Regression Analysis: Testing With 2 Variable, 2 Level Factorial Design



Response Percent Clearance

Whole Model

Summary of Fit

RSquare	0.999833
RSquare Adj	0.999828
Root Mean Square Error	0.0008
Mean of Response	0.624092
Observations (or Sum Wgts)	120

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-0.004575	0.004747	-0.96	0.3371
Probability of Classification	0.35525	0.005888	60.34	<.0001*
Probability of Neutralization	0.3565833	0.005888	60.57	<.0001*
Probability of Classification*Probability of Neutralization	0.0925	0.007303	12.67	<.0001*

Sorted Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Probability of Neutralization	0.3565833	0.005888	60.57	<.0001*
Probability of Classification	0.35525	0.005888	60.34	<.0001*
Probability of Classification*Probability of Neutralization	0.0925	0.007303	12.67	<.0001*

Note that the estimated coefficients in Figure 85 match (with minimal error due to the introduced variability) the true system performance presented in Equation (6). The probability of classification is estimated as 0.355 and probability of neutralization is estimated as 0.356 and the interaction between the variables is estimated as 0.0925; confidence intervals for all three coefficients include the actual values of 0.35, 0.35, and 0.10. The regression techniques employed on the data collected for the test configurations specified by the "baseline followed by excursions" approach are exactly the same as the regression techniques employed on the data collected for the test configurations specified by the factorial design approach. However, the coefficients are only estimated correctly when the test configurations are specified by an appropriate experimental design.

The above example provided an example of the most basic experimental design technique, a two-level full factorial design. Montgomery (2012) defines factorial designs as designs where, "in each complete trial or replication of the experiment all possible

combinations of the levels of the factors are investigated." By framing the problem in this way, it is possible to calculate the total number of design points that are required to completely explore all possible combinations of input factors using a factorial design. In the example shown above (two factors, each at two levels) the total number of runs is calculated by: 2×2=4 total runs. For a slightly larger design (three factors, two levels) the total number of runs is calculated by 2×2×2=8 total runs. In general, the number of design points required for a factorial design can be calculated (for k factors, each at m levels) as $m \times m \times m \times \ldots \times m = m^k$, and the designs are typically referred to as m^k factorial or m^k full factorial designs.

B. TRADITIONAL VERSUS SIMULATION EXPERIMENTS

This focus on correctly revealing the underlying relationship between inputs and outputs is one of the major reasons that experimental design is preferred to a "baseline followed by excursions" approach. In the example presented, the baseline-excursion approach failed to account for the potential interaction between the probabilities of classification and neutralization. Utilization of a factorial design ensured that this interaction could be correctly estimated through regression. While the value of examining all possible combinations of variables is evident from the example, it may not be apparent why factorial designs are not appropriate for examining large scale, complex systems, especially those tested in a simulation model. As mentioned, the previous example used a 2^k factorial design. It is often necessary to examine system components at more than two levels. However, increasing the number of levels for multiple components quickly renders factorial designs inappropriate for use in examining large scale, complex systems. Specifically, the total number of design points required to conduct a factorial design becomes untenable. A more detailed example based on the LCS MCM system presented in this research is illustrative of this challenge.

In the example presented in the previous section the true model performance was defined by Equation (6), which described a system where the percent clearance was impacted in a linear manner by the probability of classification, the probability of neutralization, and the interaction between those variables. Utilization of a 2^k factorial

design was sufficient to describe the performance of that system. However, when the true system behavior becomes more complicated, it may be necessary to test at an increased number of levels. Estimating a quadratic effect requires a minimum of three levels, estimating a cubic effect requires a minimum of four levels, etc. This need to move beyond 2^k factorial designs, as well as the need to examine more than two variables, quickly leads to issues with factorial designs.

As shown in Chapter IV, the LCS MCM performance is impacted by many factors, such as: the probability of detection, the probability of identification, the probability of reacquisition, the search speed, the transit speed to the minefield, the percentage of the minefield searched by surface assets, etc. If the engineer wants to investigate the impact of these six factors, along with the probability of classification and neutralization, through a 3^k factorial design the engineer would need to investigate 3^8=6,561 different design points. This dramatic increase in the required number of design points is the primary reason that factorial designs are unsuitable for investigating large scale, complex systems. Because factorial designs explicitly investigate each input variable of interest, as well as all of the interactions between these input variables, eventually factorial designs become unsuitable for examination of large scale, complex systems (as well as large scale, complex system simulation models). Table 17 summarizes the total number of design points required to conduct a factorial design based experiment for different numbers of factors and levels.

Table 17 Number of Runs Required: Full Factorial Designs

		Number of Levels			
		2	*3*	*4*	*5*
Number of Factors	*1*	2	3	4	5
	2	4	9	16	25
	3	8	27	64	125
	4	16	81	256	625
	5	32	243	1,024	3,125
	6	64	729	4,096	15,625
	7	128	2,187	16,384	78,125
	8	256	6,561	65,536	390,625
	9	512	19,683	262,144	1,953,125
	10	1,024	59,049	1,048,576	9,765,625
	11	2,048	177,147	4,194,304	48,828,125
	12	4,096	531,441	16,777,216	244,140,625
	13	8,192	1,594,323	67,108,864	1,220,703,125
	50	1.1259E+15	7.17898E+23	1.26765E+30	8.88178E+34
	100	1.26765E+30	5.15378E+47	1.60694E+60	7.88861E+69

Table 17 shows why factorial designs are inappropriate for testing the performance of large scale, complex systems. Sanchez and Wan (2012) present a powerful example demonstrating the incredible number of runs associated with full factorial designs. Referencing the IBM "Sequoia" supercomputer, which is capable of 16 petaflops (a single petaflop is a quadrillion operations per second), they note that it would require over 2.5 million years to conduct an investigation of a 2^k, 100 variable design (defined in Table 17 as approximately 1.26×10^{30} design points). In the case of large scale, complex systems, the use of traditional factorial experimental designs to evaluate the performance of various system characteristics is unreasonable. Other experimental design techniques must be considered. Figure 48 in Chapter III provided a summary of appropriate types of experimental designs that should be used in different situations and recommended the use of nearly orthogonal, balanced designs for investigation of large scale, complex systems through simulation models. Per the guidance presented earlier, both Sanchez and Wan (2012) and Vieira et al., (2013) provide concise explanations of the power of those designs, as well as instructions regarding their development and

implementation. These designs are available at harvest.nps.edu as well as overview presentations that provide detailed guidance on their implementation and application.

As mentioned, if the engineer from the example presented earlier wanted to investigate eight variables at three levels each using a full factorial design in an LCS MCM simulation model, 3^8=6,561 test configurations would need to be tested. Conducting 30 replications of each test point would therefore require 6,561×30=196,830 tests. Space filling designs have been developed that overcome the computational limitations of factorial designs while continuing to provide excellent coverage throughout the design space. In particular, the NOLH and NO/B designs recommended by this research can be developed to handle any number of factors at any number of levels. As an example, the eight variables mentioned above could be explored using an NOLH with 33 test configurations instead of 6,561, and the computational savings increase as the number of factors increases. The NOLH designs are created with an emphasis on ensuring minimal correlation between factors (that is, an increase in one factor is not associated with an increase in a second factor). The NO/B designs also ensure a relatively equal number of design points at each level for each discrete-valued factor.

While this simple example does not fully cover the capabilities and limitations of space filling designs (see Sanchez and Wan 2012 for a more complete discussion on the application of space filling designs for simulation experiments), it should provide a general overview of the intended utility of such designs and demonstrate their applicability in scenarios with a large number of potential factors, each of which must take multiple levels, which makes it impractical or improbable to use traditional factorial designs.

APPENDIX C. INNOSLATE ARCHITECTURE IMPLEMENTATION

This appendix presents an alternative representation of the mine warfare architecture products detailed in the body of the dissertation. This demonstrates that the development of the architecture is possible within multiple tools, and also overcomes a limitation associated with the current implementation of the methodology in Vitech CORE. While Vitech CORE is a powerful tool that enforces consistency between architecture products and facilitates rapid generation and iteration of those products, there are limitations associated with development of an executable architecture within CORE that can check for logical consistency within the architecture. Recall that each SysML Diagram required a "connected" structure. That is, any elements created within a diagram needed to be contained and utilized within that diagram. For many cases this is not an issue, however it is often necessary to create control type elements to represent decisions that are made at different levels of the organizational structure that cannot be represented within a single, connected diagram. As an example, the MCM-1 Avenger and the LCS do not choose which system will be utilized in a given operation; this is done by a higher level command and control element. This can be represented within the SysML Diagrams created in CORE, however it requires the user to integrate a command and control output (the decision to use either the MCM-1 Avenger or the LCS) into a decision loop for the Active, Defensive MCM Operations. The implementation of such a disconnected decision is notionally possible within CORE, but requires scripting and abstracting of system elements that removes potentially valuable information and element characteristics from visibility on the diagram itself. Re-implementing the diagrams using Innoslate's LML Action Diagrams allows a user to execute activities even when a nested alternative requires representation at another level of the system physical hierarchy or organizational structure. While CORE is capable of utilizing a "kill" setting for AND branches to provide a similar capability, that setting is only applicable to concurrent (rather than alternate) branches, limiting the applicability in this specific instance. Note that this limitation is not exclusive to CORE, many software architecting tools assume that control is transferred linearly and while they facilitate representation of this type of

structure, the creation of an executable architecture that checks for logical consistency does not support those types of definitions. This appendix demonstrates that the utilization of an alternative system architecting tool (Innoslate) allows a user to mirror the creation of the system architecture diagrams created in CORE and, due to the increased capabilities of the software, run a simulation of the system architecture to check for logical consistency. This appendix will walk through a series of diagrams which mirror the activity diagrams presented earlier in this dissertation. It will then show the results of an execution of the activities to demonstrate that the logical structure is consistent. The activities have been associated with durations and probabilities and the execution duration approximates the duration (17.42 days) of scenarios modeled in ExtendSim (an average of 19 days) when the system conducts a single minefield pass.

This appendix begins with Figure 86, which presents a representation of the highest level activity, Mine Warfare Operations. Note that it is decomposed by Active Defensive MCM Operations as well as an Environmental Feedback activity and a Command and Control activity.

Figure 86 Mine Warfare Operations Activities (Innoslate Representation)

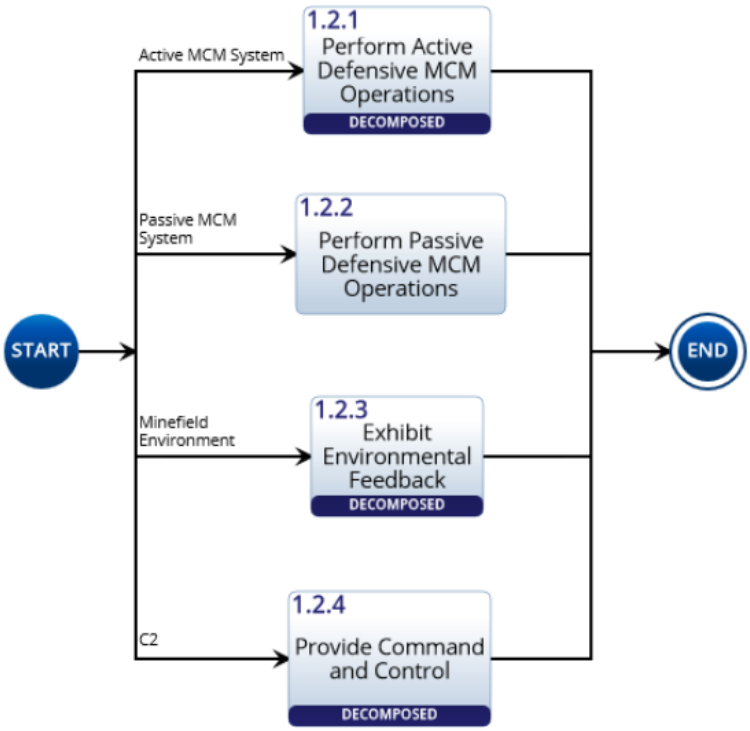


Figure 87 and Figure 88 present decompositions of the Environmental Feedback and Command and Control activities, respectively. Note that the Environmental Feedback activity produces and entity termed "Potential Mines" and the Command and Control activity produces an entities termed "Instruction to Use MCM-1 Avenger" and "Instruction to Use Littoral Combat Ship."

Figure 87 Exhibit Environmental Feedback Activities (Innoslate Representation)

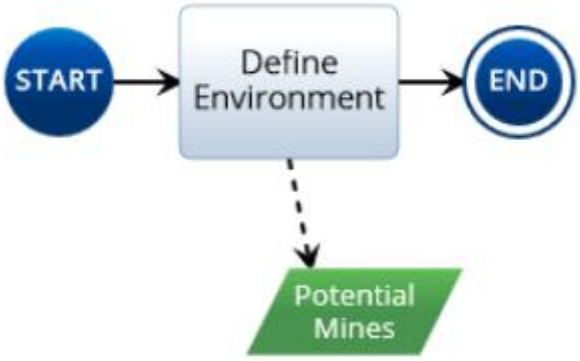
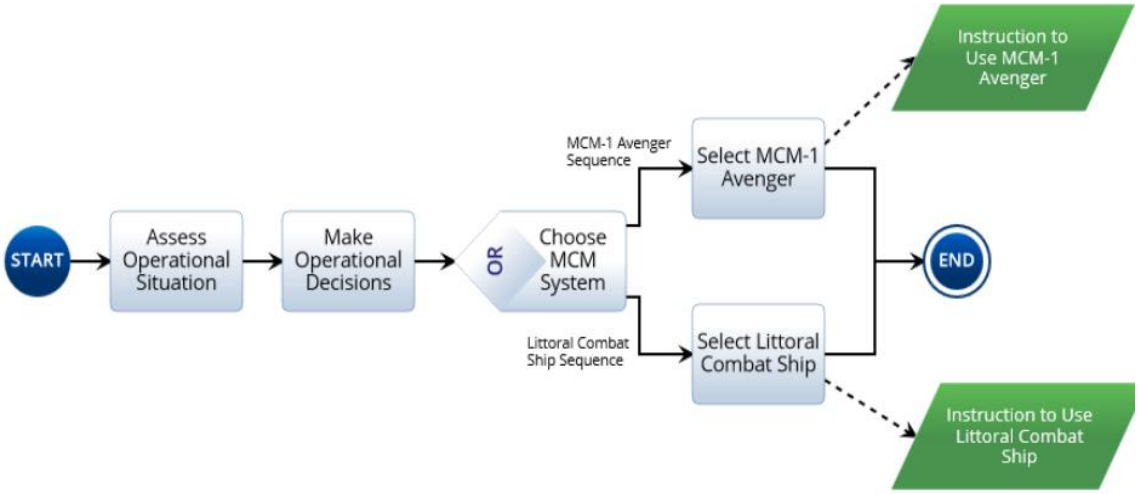


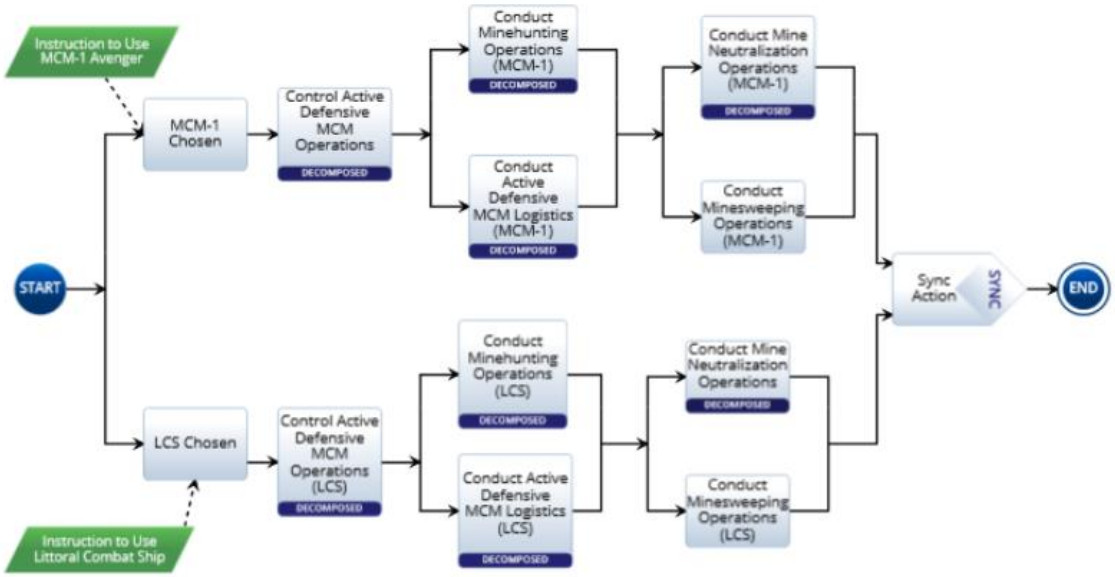
Figure 88 Provide Command and Control Activities (Innoslate Representation)



As mentioned, the advantage that Innoslate offers as an architectural software program is the ability to transfer these entities to alternative levels of the system architecture. While this can be done in CORE, it creates inconsistencies when the architecture is executed. Innoslate's representation allows the architecture to transfer these entities between levels. Figure 89 presents a decomposition of Active Defensive

MCM Operations, where the "Instruction to Use MCM-1 Avenger" and the "Instruction to Use Littoral Combat Ship" are also represented and are serving as triggers to subsequent functions.

Figure 89 Perform Active Defensive MCM Operations Activities (Innoslate Representation)



This use of these entities at different levels of decomposition would result in an error message if implemented in CORE, but is possible using Innoslate's representation. Note that the "Potential Mines" created by the Environmental Feedback activity are not utilized until several additional levels of decomposition have been explored. The "Potential Mines" are used in the Conduct Minehunting Operations activity decomposition, which is decomposed by Detect Mines, which is utilizes the "Potential Mines." Figure 90 and Figure 91 present these additional levels of decomposition.

Figure 90 Conduct Minehunting Operations Activities (Innoslate Representation)

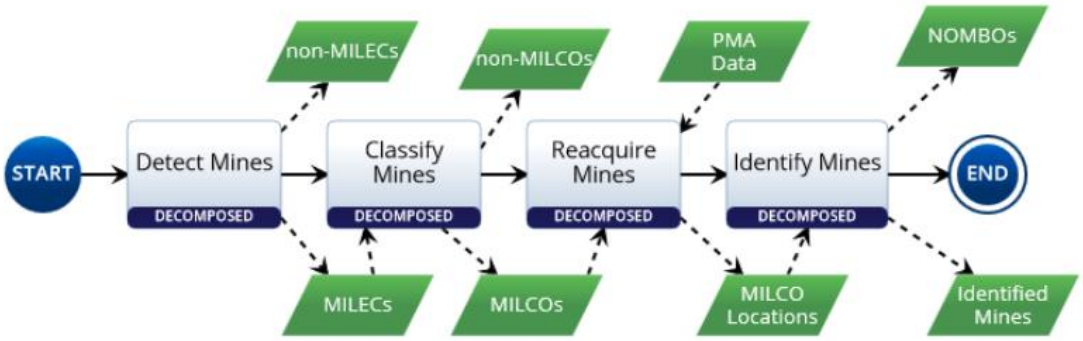
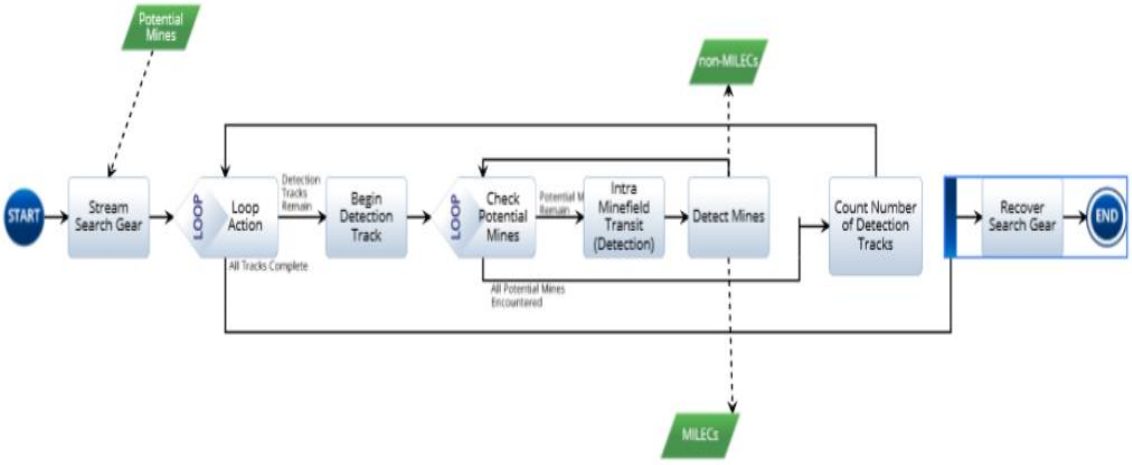
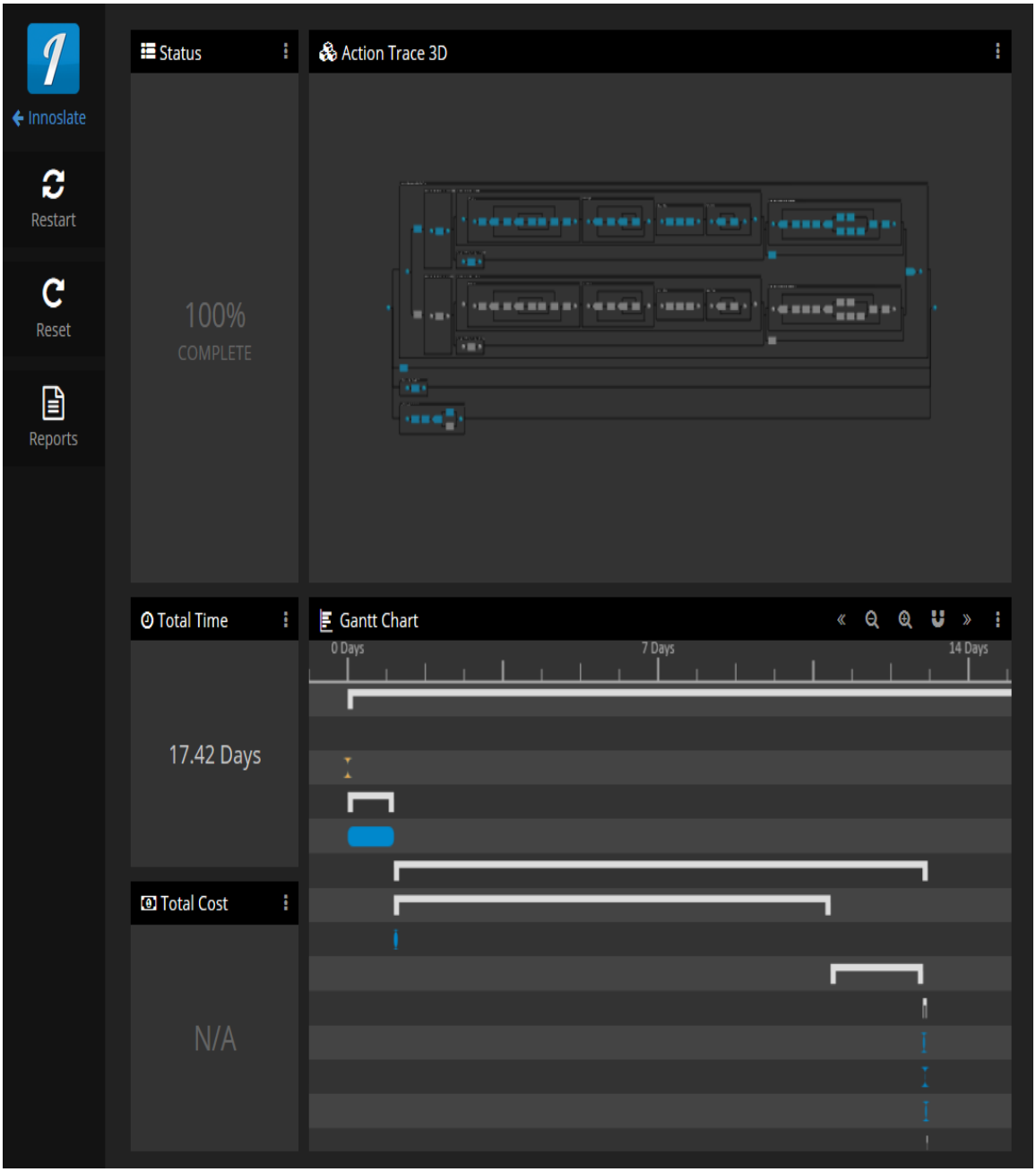


Figure 91 Detect Mines Activities (Innoslate Representation)



Note that each entity created in Innoslate must be utilized by another entity at some level of decomposition. If the logical structure is consistent and each activity is associated with a duration (and probable path, as necessary), the architecture may be executed. Figure 92 presents the results of an example execution, which approximates the results of the ExtendSim implementation of the same processes (17.42 days in Innoslate, approximately 19 days in ExtendSim).

Figure 92 Detect Mines Activities (Innoslate Representation)



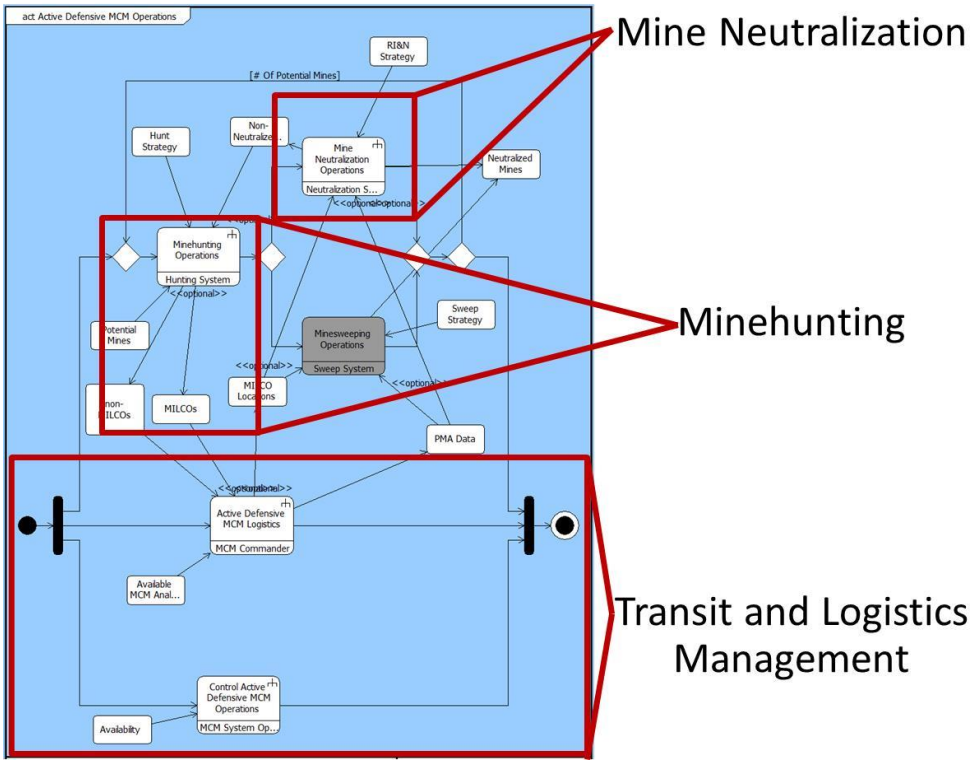
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. MODEL IMPLEMENTATION IN EXTENDSIM

This appendix presents a series of annotated figures (screenshots of architecture models and discrete event models) that demonstrate the development of a discrete event model (in the ExtendSim software) based on the architecture products presented throughout the simulation. As a point of emphasis, this appendix is not a tutorial on ExtendSim or discrete event modeling; rather it provides a visualization guiding the implementation of architecture products in an external simulation model. Note that this section does not provide a roadmap for simulation development.

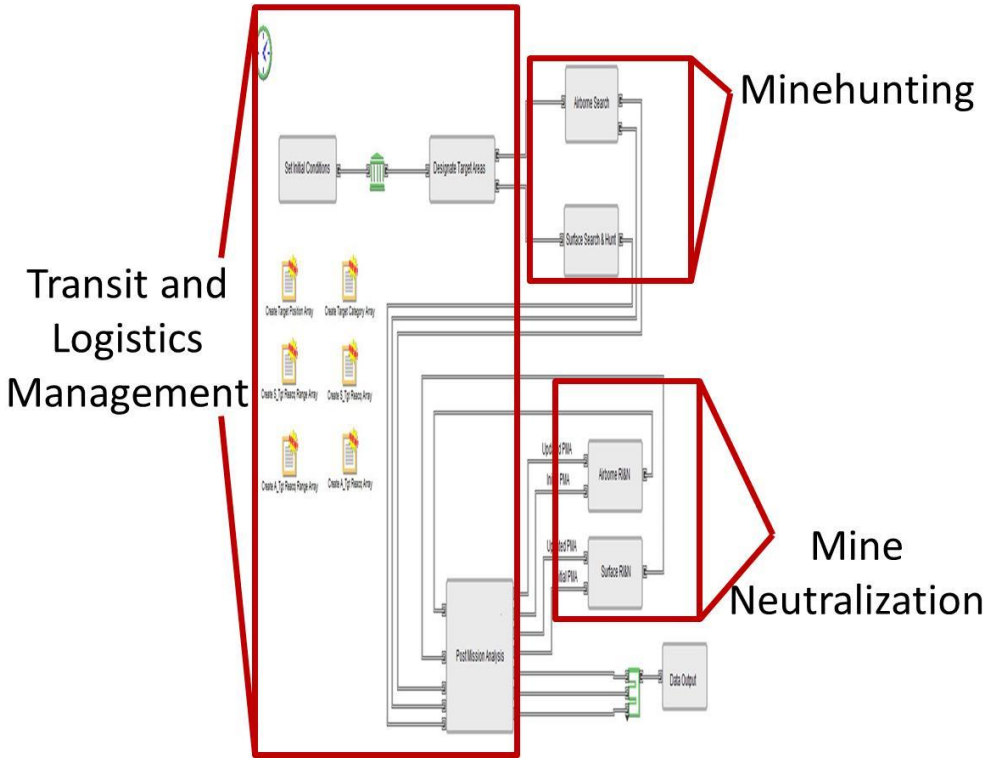
As mentioned in Chapter IV, the Activity Diagram for Active, Defensive MCM Operations suggested that the external simulation model represent three distinct activities. Figure 93 provides an annotated version of that Activity Diagram.

Figure 93 Annotated Activity Diagram: Active, Defensive MCM Operations



Development of a framework for a discrete event simulation that captures each of the three major elements of the operation occurs per the guidelines established in the Active, Defensive MCM Operations Activity Diagram. Figure 94 presents an annotated screenshot of ExtendSim, highlighting the portions of the discrete event model that correspond to each element of the Active, Defensive MCM Operation per Figure 93. This appendix uses the simulation model for the MCM-1 Avenger configurations for brevity and consistency. A similar procedure for the LCS configurations produced similar mappings.

Figure 94 Annotated Implementation of Active, Defensive MCM Operations in ExtendSim



Note that the transit and logistics management functions are implemented throughout the simulation model, first by setting initial conditions, then by designating a target area, and in between minehunting and mine neutralization by conducting post mission analysis. Minehunting and Mine Neutralization are implemented for both

airborne and surface assets. Following the general convention presented in the body of the dissertation, it is possible to decompose the Minehunting function (both in architecture products and in the simulation model) into Mine Detection and Mine Classification to visualize the mapping from the architecture models to the external simulation model. Figure 95 presents an annotated Activity Diagram for Mine Detection and Figure 96 presents an annotated screenshot of the Mine Detection events within the simulation model. Note that the SysML Activity diagram begins with a choice of either the MCM-1 Avenger Configuration or the LCS Configuration. Only the MCM-1 Avenger portion is annotated and Figure 96 only shows the ExtendSim implementation of the MCM-1 Avenger configuration.

Figure 95 Annotated Activity Diagram: Detect Mines

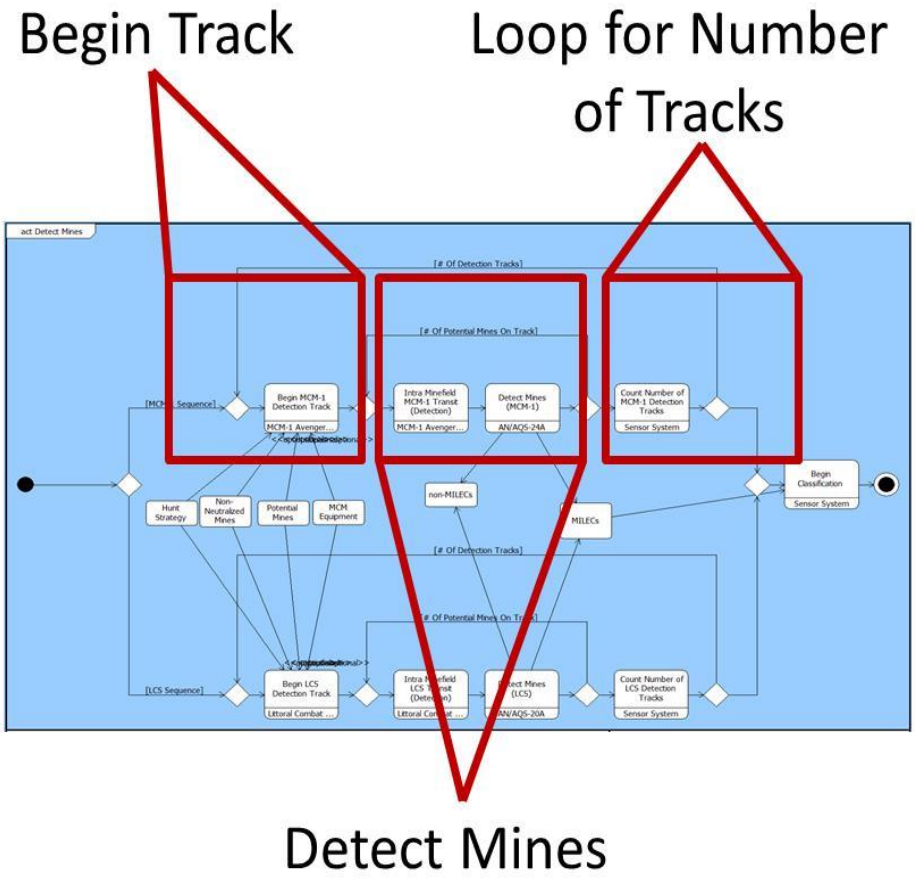
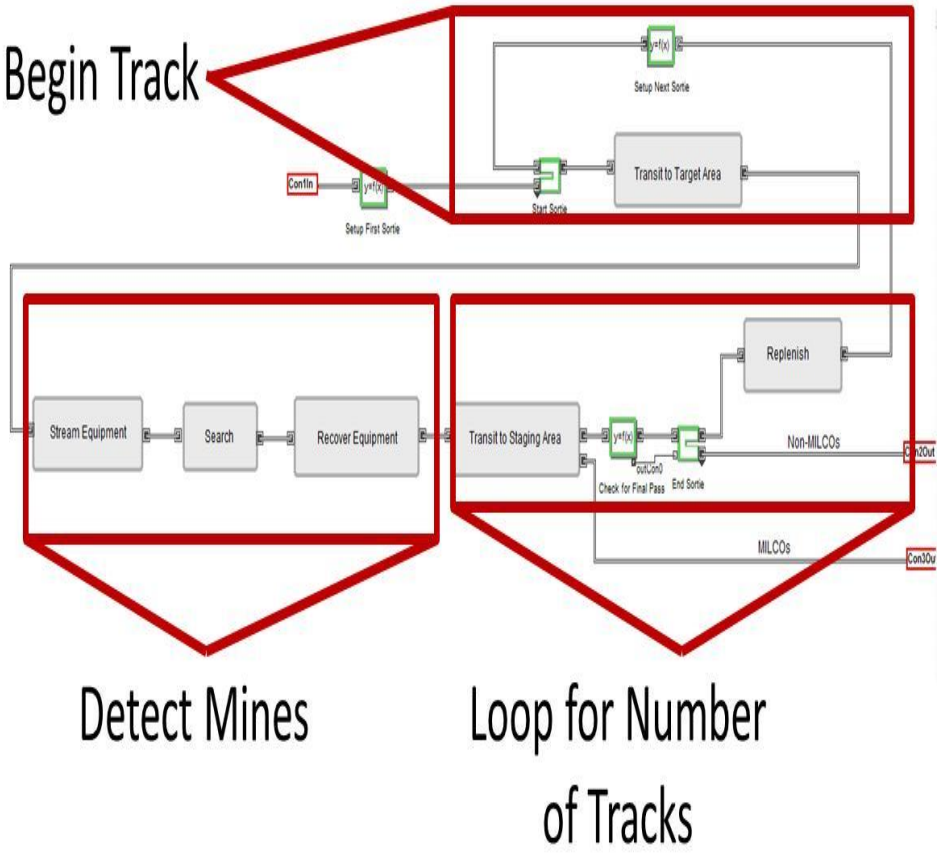


Figure 96 Annotated Implementation of Detect Mines in ExtendSim



Notice that three major phases comprise both the SysML Activity Diagram and the ExtendSim screenshot: Begin Track, Detect Mines, and Loop for Number of Tracks. ExtendSim implements the activities associated with each phase as discrete events. A distribution is assigned to each event and varied between simulation runs. A similar visualization is possible for Mine Classification. Figure 97 presents a SysML Activity Diagram for Mine Classification and Figure 98 presents an ExtendSim implementation of Mine Classification. Note once again that only the MCM-1 Avenger configuration implementation is presented and annotated.

Figure 97 Annotated Activity Diagram: Classify Mines

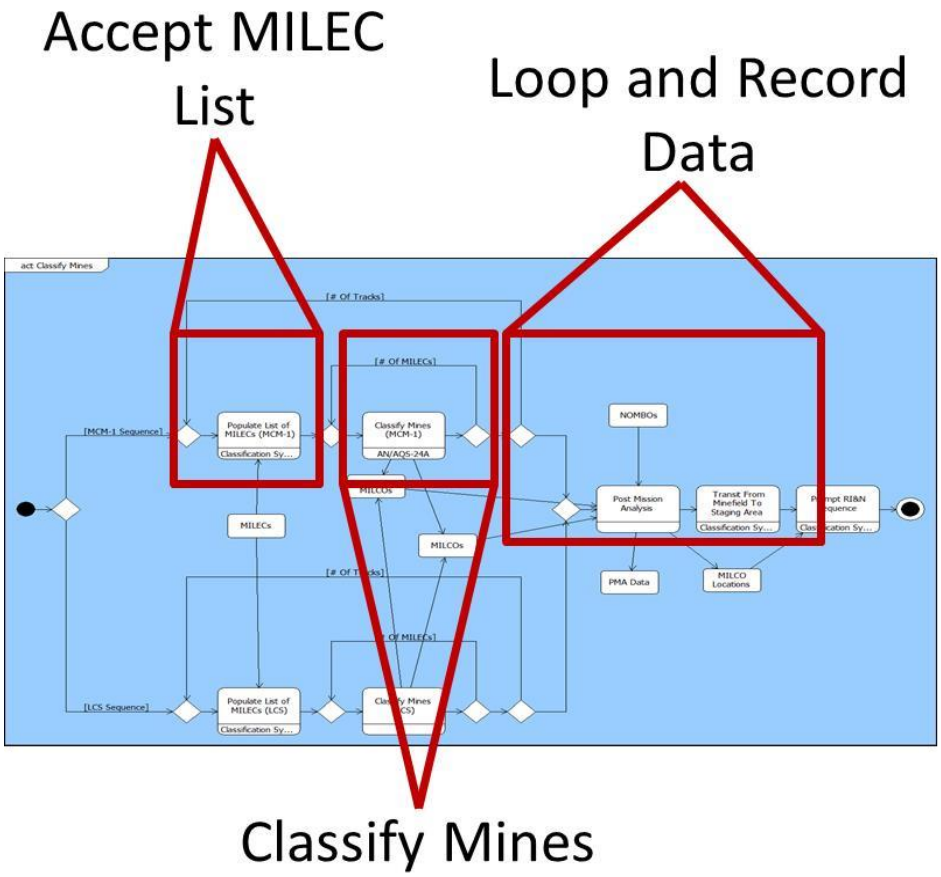
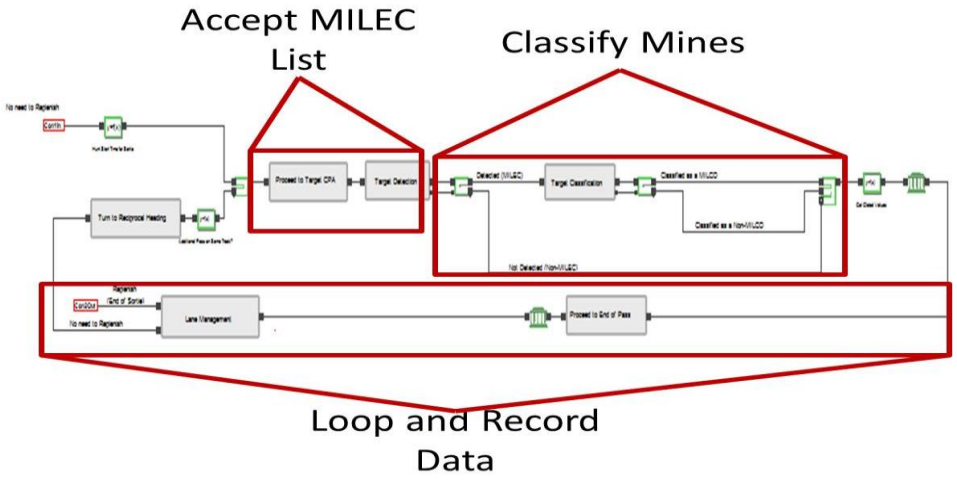


Figure 98 Annotated Implementation of Classify Mines in ExtendSim



As with Mine Detection, three major phases define Mine Classification, both within the SysML Activity Diagram and the ExtendSim implementation. The three major phases shown are: Accept MILEC List (which is an output from Mine Detection, as shown in Figure 95 and Figure 96 and highlighted during the discussion of Sequence Diagrams in Chapter III), Classify Mines, and Loop and Record Data. Once again, ExtendSim implements each of the activities associated with each phase as a discrete event and varies the characteristics of that event between simulation runs.

APPENDIX E. SUPPORTING ANALYSIS AND FIGURES

This section provides supporting analysis and figures not deemed necessary for presentation in the body of the dissertation but may be of interest for review of the details of some analysis presented previously. Figure 99–Figure 101 provides initial analysis results for MCM-1 configurations while Figure 102–Figure 105 provides initial analysis results for LCS configurations.

A. SUPPORTING ANALYSIS PRODUCTS (MCM-1 CONFIGURATIONS)

This section presents the regression analysis referenced in Chapter IV. This regression analysis is the basis for the surrogate models used in the tradespace visualizations in Chapter IV.

Figure 99 Regression Analysis: Percent Clearance (MCM-1 Configurations)

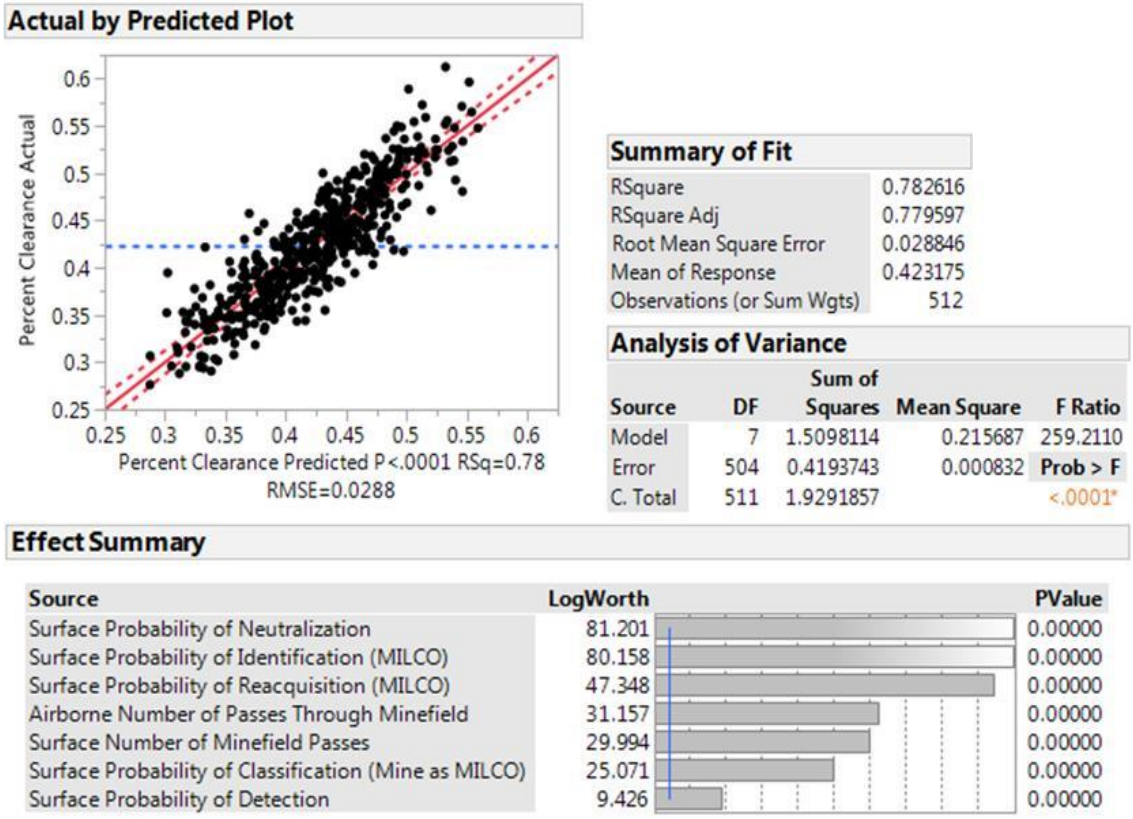


Figure 100 Regression Analysis: Probability of 90% Detection (MCM-1 Configurations)

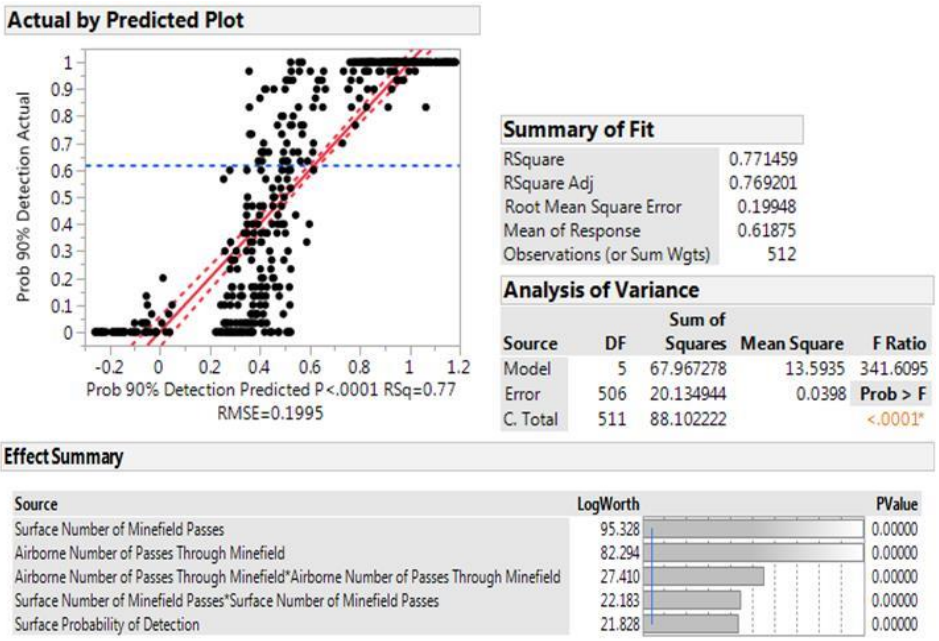
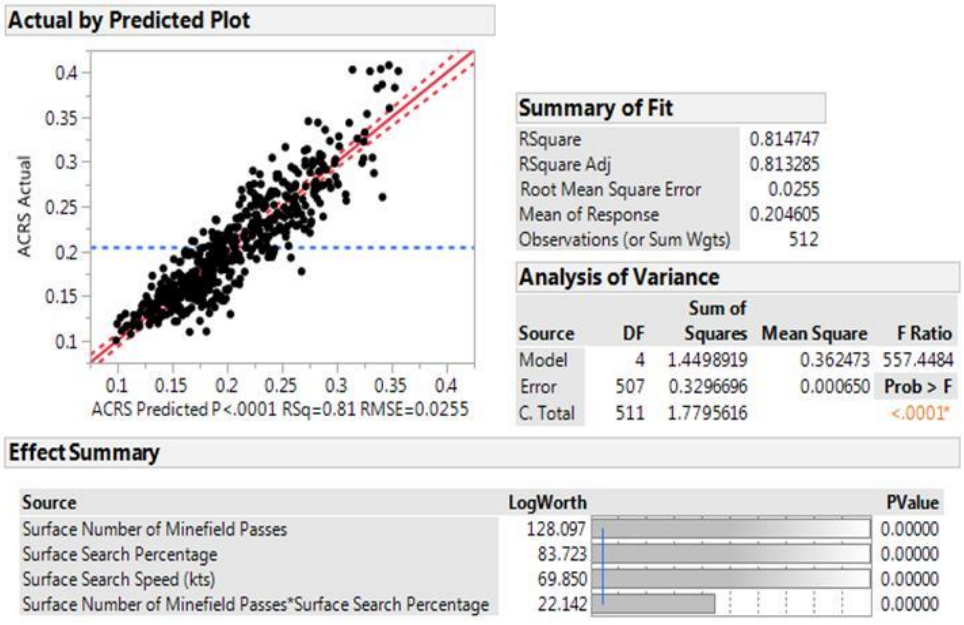


Figure 101 Regression Analysis: Area Coverage Rate Sustained (MCM-1 Configurations)



B. SUPPORTING ANALYSIS PRODUCTS (LCS CONFIGURATIONS)

This section presents the regression analysis for the LCS MCM Configurations referenced in Chapter IV. This regression analysis is the basis for the surrogate models used in the tradespace visualizations in Chapter IV. Note that the regression model for the Probability of 90% Detection suggested two distinct groupings in the data; therefore a Partition Tree analysis is used as an alternative to regression analysis.

Figure 102 Regression Analysis: Percent Clearance (LCS Configurations)

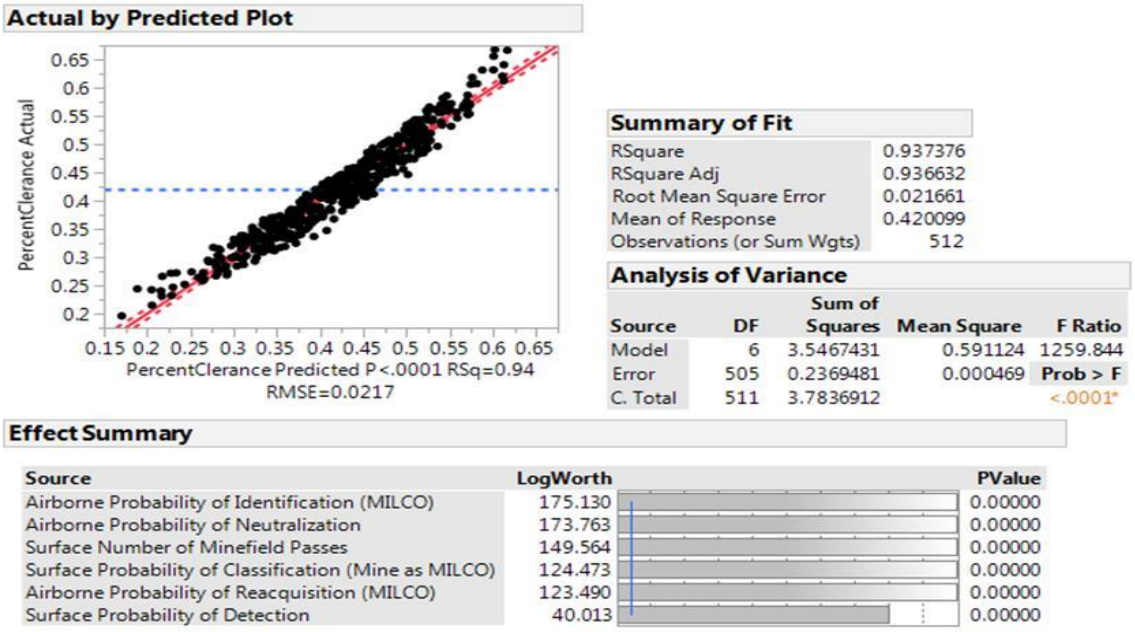
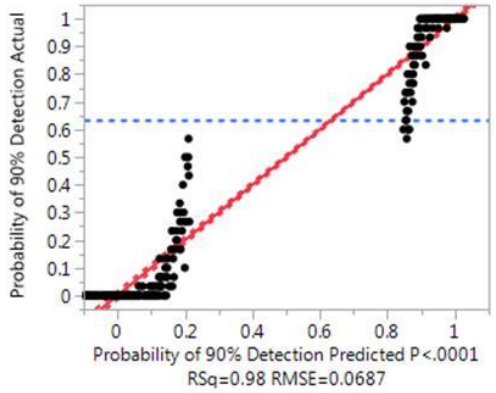


Figure 103 Regression Analysis: Probability of 90% Detection (LCS Configurations)

Actual by Predicted Plot



Summary of Fit

RSquare	0.976909
RSquare Adj	0.976727
Root Mean Square Error	0.06872
Mean of Response	0.633464
Observations (or Sum Wgts)	512

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	4	101.29240	25.3231	5362.338
Error	507	2.39426	0.0047	Prob > F
C. Total	511	103.68666		<.0001*

Effect Summary

Source	LogWorth	PValue
Surface Number of Minefield Passes	403.564	0.00000
Surface Number of Minefield Passes*Surface Number of Minefield Passes	239.303	0.00000
Surface Probability of Detection	56.700	0.00000
Surface Number of Minefield Passes*Surface Probability of Detection	22.393	0.00000

Figure 104 Regression Analysis: Area Coverage Rate Sustained (LCS Configurations)

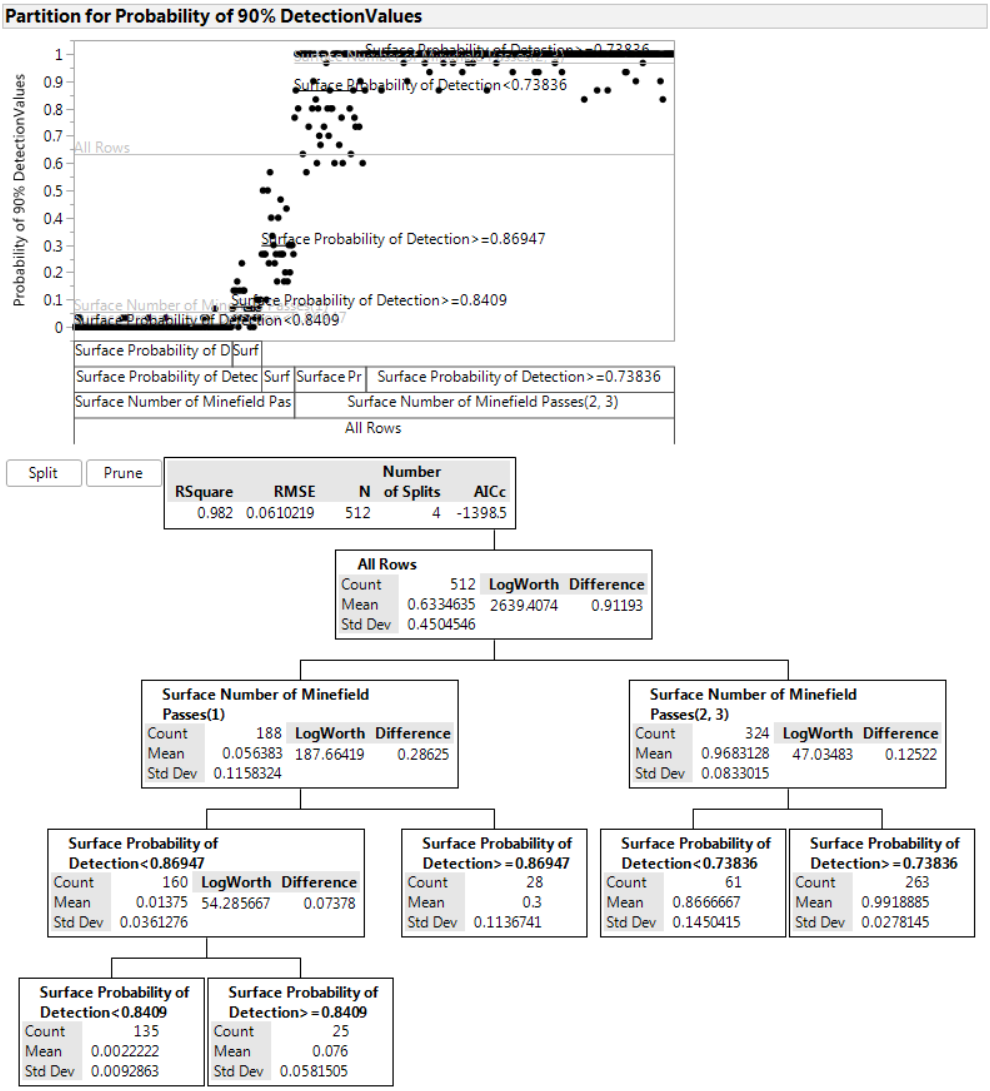
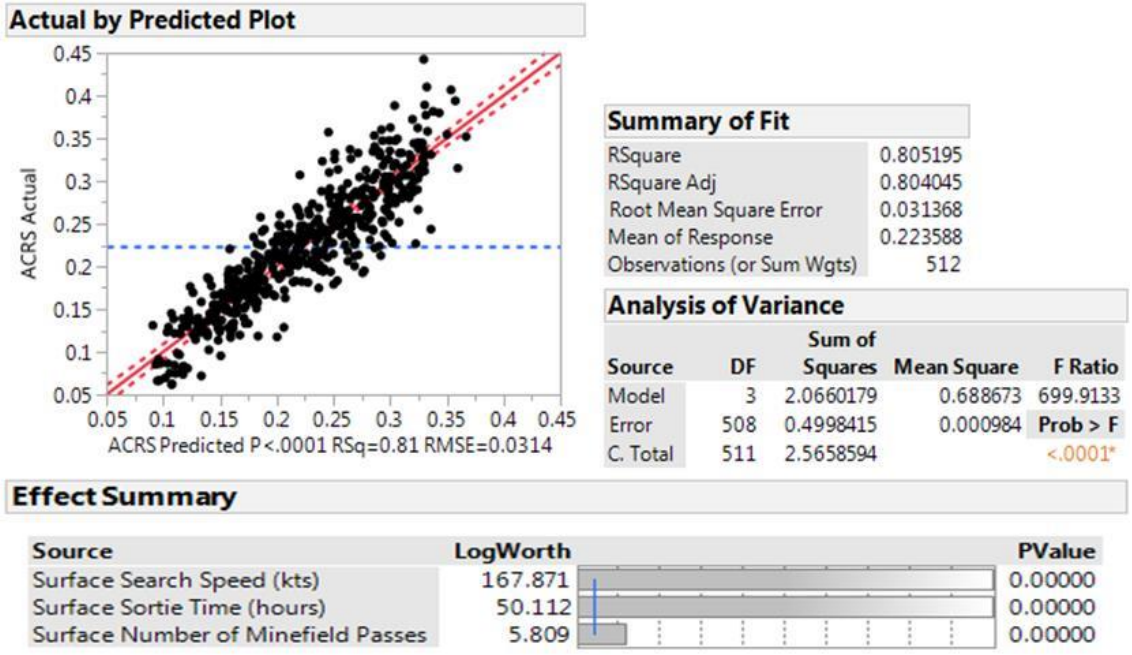


Figure 105 Regression Analysis: Area Coverage Rate Sustained (LCS Configurations)



THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Acheson, Paulette, Cihan Dagli, and Nil Kilicay-Ergin. 2013. "Model Based Systems Engineering for System of Systems Using Agent-Based Modeling." In *Procedia Computer Sciences*: Vol. 16, edited by Christiaan J.J. Paredis, Carlee Bishop, and Douglas Bodner, 11–19. Atlanta, GA: Elsevier.

Amador, Brian. 2011. "U.S. Navy Funding Goals for Future Mine Warfare Capability." Lecture at the 16th Annual Expeditionary Warfare Conference, Panama City, FL.

Ashpari, Mohammad J. 2012. "A Capability-Based Approach to Analyzing the Effectiveness and Robustness of an Offshore Patrol Vessel in the Search and Rescue Mission." Master's Thesis. Naval Postgraduate School.

Balestrini-Robinson, Santiago, Dane F. Freeman, and Daniel C. Browne. 2015. "An Object-Oriented and Executable SysML Framework for Rapid Model Development." In *Procedia Computer Sciences*: Vol. 44, edited by Jon Wade and Robert Cloutier, 423–432. Hoboken, NJ: Elsevier.

Becker, Nicole, Timothy Byram, David Frank, Kevin Hogan, Richard Kim, Glenna Miller, Shane Schonhoff, Scott Myers, and Heather Whitehouse. 2014. "Application of Model-Based Systems Engineering (MBSE) to Compare Legacy and Future Forces in Mine Warfare (MIW) Missions." Capstone Report. Naval Postgraduate School.

Behdani, Behzad. 2012. "Evaluation of Paradigms for Modeling Supply Chains as Complex Socio-Technical Systems." In *Simulation Conference (WSC), Proceedings of the 2012 Winter Simulation Conference*, edited by Christoph Laroque, Jan Himmelspach, Raghu Pasupathy, Oliver Rose, and Adelinde Uhrmacher, 1–15. Berlin: IEEE.

Bjorkman, Eileen A., Shahram Sarkani, and Thomas A. Mazzuchi. 2013. "Using Model-Based Systems Engineering as a Framework for Improving Test and Evaluation Activities." *Systems Engineering* 16 (3): 346–362.

Blanchard, Benjamin S., and Wolter J. Fabrycky. 2010. *Systems Engineering and Analysis*, 5th ed. Upper Saddle River, NJ: Pearson Prentice Hall

Boehm, Barry. 1986. "A Spiral Model of Software Development and Enhancement." *ACM SIGSOFT Software Engineering Notes* 11(4): 14–24.

Buede, Dennis, M. 2009. *The Engineering Design of Systems, Models and Method*, 2nd Edition. New York, NY: John Wiley & Sons

Carpenter, Wendi B. 2010. *Navy Warfare Publication: Naval Mine Warfare*. Vol. 1. NWP 3–15. Norfolk, VA: Navy Warfare Development Command.

Carson, Ronald S., and Barbara J. Sheeley. 2013. "Functional Architecture as the Core of Model-Based Systems Engineering." In *INCOSE International Symposium* 23, 29–45. Philadelphia, PA: INCOSE.

Department of Defense. 1974. *Engineering Management.* MIL-STD-499A. Washington, DC: Department of Defense.

Department of Defense. 1993. *Systems Engineering.* MIL-STD-499B. Washington, DC: Department of Defense.

Department of Defense Chief Information Officer. 2015. "DoDAF: DOD Architecture Framework Version 2.02 DOD Deputy Chief Information Officer." August 11. http://dodcio.defense.gov/Library/DoDArchitectureFramework.aspx

Dori, Dov. 2002. *Object Process Methodology: A Holistic Systems Paradigm*. New York: Springer.

Dori, Dov, Iris Reinhartz-Berger, and Arnon Sturm. 2003. "Developing Complex Systems with Object-Process Methodology using OPCAT." In *Proceedings of the 22nd International Conference on Conceptual Modeling*, edited by Il-Yeol Song, Stephen W. Liddle, Tok-Wang Ling, and Peter Scheurmann ,570-572. Chicago, IL: Springer-Verlag.

Ellman, Jesse E. 2009. "The Role of Evolutionary Acquisition and Spiral Development in the Failure of the Army's Future Combat System." Master's Thesis. Georgetown University.

Emes, Michael, Peter Bryant, Mike Wilkinson, Paul King, Ady James, and Stuart Arnold. 2012. "Interpreting 'Systems Architecting.'" *Systems Engineering* 15(4): 369–395.

Estefan, Jeff A. 2008. *Survey of Model-Based Systems Engineering (MBSE) Methodologies,* Rev B. Pasadena, CA: California Institute of Technology.

Farr, John V. 2011. *Systems Life Cycle Costing: Economic Analysis, Estimation, and Management*. Boca Raton, FL: CRC Press.

Fisher, Amit. 2013. "IBM System and Software Solutions: Design and Model Management across the Product Development Life cycle." Presentation at the INCOSE 2013 MBSE Workshop, Jacksonville, FL, January 26–27.

FHWA Operations. 2013. "Systems Engineering for ITS Handbook - Section 3 What Is Systems Engineering." Dec 9. http://ops.fhwa.dot.gov/publications/seitsguide/section3.htm

Florida Department of Transportation. 2003. *A Process Review and Appraisal of the Systems Engineering Capability for the Florida Department of Transportation (FDOT).* Technical Memorandum No. 1. Tallahassee, FL: Florida Department of Transportation.

Foorsberg, Kevin, Hal Mooz, and Howard Cotterman, 2005. *Visualizing Project Management: Models and Frameworks for Mastering Complex Systems*. New York, NY: John Wiley & Sons.

Friedenthal, Sanford., Regina Griego, and Mark Sampson. 2007. "INCOSE Model-based Systems Engineering (MBSE) Initiative." Presented at the INCOSE 2007 Symposium, San Diego, CA, June 24–28.

Friedenthal, Sanford., Alan Moore, and Rick Steiner. 2009. *A Practical Guide to SysML The Systems Modeling Language.* San Francisco, CA: Morgan Kaufmann.

Garrett, Robert K., Steve Anderson, Neil T. Baron, and James D. Moreland, Jr. 2011. "Managing the Interstitials, a System of Systems Framework Suited for the Ballistic Missile Defense System." *Systems Engineering* 14(1): 87–109.

Ge, Bingfeng, Keith W. Hipel, Kewei Yang, and Yingwu Chen. 2013. "A Data-Centric Capability-Focused Approach for System-of-Systems Architecture Modeling and Analysis." *Systems Engineering* 16(3): 363–377.

Giammarco, Kristin, and Mikhail Auguston. 2013. "Well, You Didn't Say Not to! A Formal Systems Engineering Approach to Teaching an Unruly Architecture Good Behavior." In *Procedia Computer Sciences*: Vol. 20, edited by Cihan Dagli, 277–282. Rolla, MO: Elsevier.

Hagel, Charles T. 2013. "Speech Delivered to National Defense University." Speech. Washington, DC, April 3.

Haveman, Steven P., and G. Maarten Bonnema. 2015. "Communication of simulation and modelling activities in early systems engineering." In *Procedia Computer Sciences*: Vol. 44, edited by Jon Wade and Robert Cloutier, 305–314. Hoboken, NJ: Elsevier.

Hoffman, Hans-Peter. 2011. *Model-Based Systems Engineering with Rational Rhapsody and Rational Harmony for Systems Engineering*, Release 3.1.2. Somers, NY: IBM Corporation.

Humman, James, and Azad M. Madni. 2014. "Integrated Agent-Based Modeling and Optimization in Complex Systems Analysis." In *Procedia Computer Sciences*: Vol. 28, edited by Azad M. Madni and Barry Boehm, 818–827. Malvern, PA: Elsevier.

INCOSE. 2011. *INCOSE OOSEM Working Group Charter.* San Diego, CA: INCOSE.

Kaymal, Turgut. 2013. "Assessing the Operational Effectiveness of a Small Surface Combat Ship in an Anti-Surface Warfare Environment." Master's Thesis. Naval Postgraduate School.

Kleijnen, Jack P.C., Susan M. Sanchez, Thomas W. Lucas, and Thomas M. Cioppa. 2005. "A User's Guide to the Brave New World of Designing Simulation Experiments." *INFORMS Journal on Computing*, 17(3): 263–289.

Law, Averill M. 2014. *Simulation Modeling and Analysis*, 5th Edition. New York, NY: McGraw Hill.

Law, Averill M. 2009. "How to Build Valid and Credible Simulation Models." In Simulation Conference (WSC), In *Proceedings of the 2009 Winter Simulation Conference*, edited by Ann Dunkin, Ricki Ingalls, Enver Yucesan, Manuel Rossetti, Ray Hill, and Bjorn Johansson, 24–33. Austin, TX: IEEE.

Liston, Paul, Kamil Erkan Kabak, Peter Dungan, James Byrne, Paul Young, and Cathal Heavey. 2011. "An Evaluation of SysML to Support Simulation Modeling." In *Conceptual Modeling for Discrete-Event Simulation*, edited by Stewart Robinson, Roger Brooks, Kathy Kotiadis, and Durk-Jouke van der Zee, 279–309. Boca Raton, FL: CRC Press.

Lucas, Thomas W., W. David Kelton, Paul J. Sanchez, Susan M. Sanchez, Ben L. Anderson. 2015. "Changing the Paradigm: Simulation, Now a Method of First Resort." *Naval Research Logistics* 62(4): 293–303.

MacCalman, Alexander D. 2013. "Flexible Space-Filling Designs for Complex System Simulations." Ph.D. Dissertation, Naval Postgraduate School.

MacCalman, Alex, Hyangshim Kwak, Mary McDonald, and Stephen Upton. 2015. "Capturing experimental design insights in support of the model-based systems engineering approach." In *Procedia Computer Sciences*: Vol. 44, edited by Jon Wade and Robert Cloutier, 315–324. Hoboken, NJ: Elsevier.

MacCalman, Alex, Hyangshim Kwak, Mary McDonald, Steve Upton, Coleman Grider, Robert Hill, Hunter Wood, and Paul Evangelista. 2015. *Illuminating Tradespace Decisions Using Efficient Experimental Space-Filling Designs for the Engineering Resilient Systems Architecture.* West Point, NY: Operations Research Center United States Military Academy.

MacCalman, Alexander D., Paul T. Beery, and Eugene P. Paulo. (working paper). A Systems Design Exploration Approach that Illuminates Tradespaces using Statistical Experimental Designs. *Systems Engineering*.

Maier, Mark W. and Eberhardt Rechtin. 2009. *The Art of Systems Architecting*, Third Edition. Boca Raton, FL: CRC Press.

McKeown, Jason L. 2012. "Analyzing the Surface Warfare Operational Effectiveness of an Offshore Patrol Vessel Using Agent Based Modeling." Master's Thesis. Naval Postgraduate School.

Montgomery, Douglas C. 2012. *Design and Analysis of Experiments*, 8th edition. New York, NY: Wiley.

Myers, Raymond H., Douglas C. Montgomery, and Christine M. Anderson-Cook. 2009. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, Third Edition. New York, NY: Wiley.

Neches, Robert, and Azad M. Madni. 2013. "Towards Affordably Adaptable and Effective Systems." *Systems Engineering* 16(2): 224–234.

National Institute of Standards and Technology. 1993. *Integration Definition for Functional Modeling (IDEF0).* Technical Report, Federal Information Processing Standards Publication 183. Springfield, VA: U.S. Department of Commerce.

Object Management Group. 2012. *OMG Systems Modeling Language (OMG SysML) Version 1.3*. OMG Document: ptc/2012-04-07. Needham, MA: Object Management Group.

Object Management Group. 2006. "OMG Systems Modeling Language (OMG SysML) Tutorial." Presented at the INCOSE 2006 Symposium, Orlando, FL, July 9–13.

Object Management Group. 2003. *UML for Systems Engineering*. OMG Document: ad/03-03-41. Needham, MA: Object Management Group.

Parker, Jeffrey D. 2015. "An Innovative Approach for the Development of Future Marine Corps Amphibious Capability." Master's Thesis. Naval Postgraduate School.

Piaszczyk, Chris. 2011. "Model Based Systems Engineering with Department of Defense Architectural Framework." *Systems Engineering* 14(3): 305–326.

Program Executive Office Littoral and Mine Warfare. 2008. *Standard Mine Warfare Measures of Effectiveness*. PEO LMW Instruction 3370.1A. Washington, DC: Department of the Navy.

Qamar, Ahsan, Carl During, and Jan Wikander. 2009. "Designing Mechatronic Systems, A Model-Based Perspective, an Attempt to Achieve SysML-Matlab/Simulink Model Integration." In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics* 2009 , 1306–1311. Berlin: IEEE.

Royce, Winston W. 1970. "Managing the Development of Large Software Systems." In *Proceedings of IEEE Western Electronic Shown and Convention* 26(8), 328–338, Los Angeles, CA: IEEE.

Ross, Adam M. 2003. “Multi-Attribute Tradespace Exploration with Concurrent Design as a Value-Centric Framework for Space System Architecture and Design.” Master’s Thesis, Massachusetts Institute of Technology.

Ross, Adam M., David B. Stein, and Daniel E. Hastings. 2014. “Multi-Attribute Tradespace Exploration for Survivability.” *Journal of Spacecraft and Rockets* 51.5, 1735–1752.

Russell, Mike. 2012. “Using MBSE to Enhance System Design Decision Making.” In *Procedia Computer Sciences*: Vol. 8, edited by Cihan H. Dagli, 188–193. Rolla, MO: Elsevier.

Ryan, Jessica, Shahram Sarkani, and Thomas Mazzuchi. 2013. “Leveraging Variability Modeling Techniques for Architecture Trade Studies and Analysis.” *Systems Engineering* 17(1): 10–25.

Sage, Andrew P., and James E. Armstrong, Jr. 2000. *Introduction to Systems Engineering*. New York, NY: John Wiley & Sons

Sanchez, Susan M., and Hong Wan. 2012. “Work Smarter, Not Harder: A Tutorial on Designing and Constructing Simulation Experiments.” In *Simulation Conference (WSC), Proceedings of the 2012 Winter Simulation Conference*, edited by Christoph Laroque, Jan Himmelspach, Raghu Pasupathy, Oliver Rose, and Adelinde Uhrmacher, 1–15. Berlin: IEEE.

Sanchez, Susan M., Thomas W. Lucas, Paul J. Sanchez, Christopher J Nannini, and Hong Wan. 2012. “Designs for Large-Scale Simulation Experiments, with Applications to Defense and Homeland Security.” In *Design and Analysis of Experiments: Special Designs and Applications, Volume 3* edited by Klaus Hinkelmann, 1–26. Hoboken, NJ: John Wiley & Sons.

Sanchez, Susan M. 2000. “Robust Design: Seeking the Best of All Possible Worlds.” In Simulation Conference (WSC), In *Proceedings of the 2000 Winter Simulation Conference*, edited by Jeffrey A. Joines, Russell R. Barton, Keebom Kang, and Paul A. Fishwick, 69–76. Berlin: IEEE.

Santner, Thomas J., Brian J. Williams, and William I. Notz. 2003. *The Design and Analysis of Computer Experiments*. Springer Series in Statistics.

Sargent, Robert G. 2013. “An Introduction to Verification and Validation of Simulation Models.” In Simulation Conference (WSC), In *Proceedings of the 2013 Winter Simulation Conference*, edited by Ray Hill, Michael Kuhl, Raghu Pasupathy, Seong-He Kim, and Andreas Tolk, 231–327. Washington, DC: IEEE.

SE Handbook Working Group. 2011. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, Version 3.2.2, edited by Cecilia Haskins. San Diego, CA: International Council on Systems Engineering (INCOSE)

Siebers, Peer-Olaf, Charles M. Macal, Jeremy Garnett, David Buxton, Michael Pidd. 2010. "Discrete Event Simulation is Dead, Long Live Agent-Based Simulation!" *Journal of Simulation* 4 (3): 204–210.

Sitterle, Valerie B., Dane F. Freeman, Simon R. Goerger, and Tommer R. Ender. 2015. "Systems Engineering Resiliency: Guiding Tradespace Exploration within an Engineering Resilient Systems Context." In *Procedia Computer Sciences*: Vol. 44, edited by Jon Wade and Robert Cloutier, 649–658. Hoboken, NJ: Elsevier.

Spero, Eric., Michael P Avera, Pierre E. Valdez, and Simon R Goerger. 2014. "Tradespace Exploration for the Engineering of Resilient Systems." In *Procedia Computer Sciences*: Vol. 28, edited by Azad M. Madni and Barry Boehm, 591–600. Malvern, PA: Elsevier.

Summers, Joshua D., Claudia Eckert, and Ashok Goel. 2013. "Function in Engineering Benchmarking Representations and Models." In *Proceedings of the 19th International Conference on Engineering Design (ICED13), Design for Harmonies*, Vol. 2: Design Theory and Research Methodology, edited by Udo Lindemann, Srinivasan Venkataraman, Yong Se Kim, Sang Won Lee, Yoram Reich, and Amaresh Chakrabarti, 1–16, Seoul: ICED.

Technical Operations, INCOSE. 2007. *Systems Engineering Vision 2020*. TP-2004-004-02. San Diego, CA: INCOSE

Thompson, Andrew R. 2015. "Evaluating the Combined UUV Efforts in a Large-Scale Mine Warfare Environment." Master's Thesis. Naval Postgraduate School.

Tolk, Andreas, and Taylor K. Hughes. 2014. "Systems Engineering, Architecture, and Simulation." In *Modeling and Simulation-Based Systems Engineering Handbook*, edited by Daniele Gianni, Andrea D'Amborgio, and Andreas Tolk, 11–41. Boca Raton, FL: CRC Press.

Trainor, Timothy, and Gregory S. Parnell. 2011. "Problem Definition." In *Decision Making in Systems Engineering and Management*, edited by Gregory S. Parnell, Patrick J. Driscoll, and Dale L. Henderson, 297–353. Hoboken, NJ: John Wiley & Sons.

Treml, Tobias. 2013. "A Revolutionary Approach for the Development of Future Ground Combat System Specifications." Master's Thesis. Naval Postgraduate School.

Vieira, Jr. Helcio. 2012. “NOB_Mixed_512DP_template_v1.xls design spreadsheet.” October 9. http://harvest.nps.edu

Vieira, Jr, Helcio, Susan M. Sanchez, Karl Heinz Kienitz, and Mischel Carmen Neyra Belderrain. 2013. “Efficient, nearly orthogonal-and-balanced, mixed designs: an effective way to conduct trade-off analysis via simulation.” *Journal of Simulation* 7(4): 264–275.

Vieira, Jr, Helcio, Susan M. Sanchez, Karl Heinz Kienitz, and Mischel Carmen Neyra Belderrain. 2011. “Generating and Improving Orthogonal Designs by Using Mixed Integer Programming.” *European Journal of Operations Research* 215: 629–638.

Vitech Corporation. 2011. *A Primer for Model-Based Systems Engineering*. Blacksburg, VA: Vitech Corporation.

Vitech Corporation. 2010. *CORE 7 System Definition Guide.* Blacksburg, VA: Vitech Corporation.

Wagner, David A., Matthew B. Bennett, Robert Karban, Nicolas Rouquette, Steven Jenkins, Michel Ingham. 2012. “An Ontology for State Analysis: Formalizing the Mapping to SysML.” In *2012 IEEE Aerospace Conference*, 1–16, Piscataway, NJ: IEEE

Wakeman, Clifford C. 2012. “Discrete Event Simulation Modeling and Analysis of Key Leader Engagements.” Master’s Thesis. Naval Postgraduate School.

Wang, Renzhong, and Cihan H. Dagli. 2011. “Executable System Architecting Using Systems Modeling Language in Conjunction with Colored Petri Nets in a Model-Driven Systems Development Process.” *Systems Engineering* 14(4): 383–409.

Weilkiens, Tim. 2008. S*ystems Engineering with SysML/UML: Modeling, Analysis, Design*. San Francisco, CA: Morgan Kaufmann Publishers.

Wu, Chunlong, Benjamin Ciavola, and John Gershenson. 2013. “A Comparison of Function and Affordance Based Design.” In *Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information Engineering Conference*, 1–9, Portland, OR: ASME.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California